AD-A055 161 STANFORD UNIV CALIF STANFORD ELECTRONICS LABS F/G 9/2
TECHNIQUES AND APPLICATIONS OF COMPUTER-AIDED CIRCUIT SIMULATIO--ETC(U)
FEB 74 R W DUTTON

UNCLASSIFIED SU-SEL-74-005 NL

1 OF 2

AD A055161

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DDC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

(14) SU - SEL - 74 - 005

(6) Techniques and Applications of Computer-Aided Circuit Simulation for Integrated Circuit and System Design. PART I. CAD Techniques. (1)

by

(10) Robert W. Dutton

(9) Technical Report.

(11) Feb[redacted]74

(12) 100 p.

This work was supported by the General Motors Corporation through their Research Laboratories, Warren, Michigan.

INTEGRATED CIRCUITS LABORATORY

STANFORD ELECTRONICS LABORATORIES

STANFORD UNIVERSITY · STANFORD, CALIFORNIA

332 400

SU SEL-74-005

TECHNIQUES AND APPLICATIONS OF COMPUTER-AIDED CIRCUIT SIMULATION FOR INTEGRATED CIRCUIT AND SYSTEM DESIGN

PART I: CAD TECHNIQUES

by

Robert W. Dutton

February 1974

This work was supported by the
General Motors Corporation through their
Research Laboratories, Warren, Michigan.

Integrated Circuits Laboratory
Stanford Electronics Laboratories
Stanford University Stanford, California

TABLE OF CONTENTS

FORWARD

This is the first part of a two part report dealing with Techniques and Applications of Computer-Aided Circuit Simulation. The topics discussed in Part I are the backbone of a seminar course, EE 417-2, which was given Autumn 1973. Support for the seminar from the General Motors Research Laboratories is gratefully acknowledged.

During the course of EE 417-2 the fundamentals of CAD techniques were developed. At the same time eight doctoral students used CAD program SPICE to investigate specific modeling and circuit design problems which are important in their thesis studies. These specific problems are coupled with projects supported by the NIH, JSEP and ONR.

A primary result of the seminar has been to bring together theoretical and experimental information on CAD and its application which has thus far been difficult to obtain.

Part I is a concise statement of CAD techniques with examples pertinent to SPICE. Part II reports the results of the investigations by the doctoral students involved in the seminar. The titles of the individual projects are given in the last subsection of Part I.

The seminar participants are gratefully acknowledged for their individual contributions to this report as well as for the collective feedback provided by the seminar. Special thanks goes to Mr. Peter Slapnicar and Mr. Tak Young who were major contributors to the seminar via discussions of CAD techniques and program developments.

INTRODUCTION

The need for computer simulation is steadily increasing in the design of Integrated Circuits and systems using IC's. The process dependence of component parameters and parasitic device effects cannot be adequately breadboarded to test performance for IC's. Computer-Aided Circuit Analysis provides an efficient and flexible means to evaluate the performance of circuit designs prior to the costly and time-consuming fabrication process. Component values and process-dependent device parameters can be varied to optimize circuit performance and improve fabrication yields. However, to effectively use these computer tools, the user must be able to supply the programs with realistic models and model parameters. Furthermore, a basic understanding of the techniques used for computer circuit simulation can allow the designer to critically assess the validity of the simulated results.

This report presents material and examples pertinent to the enlightened use of sophisticated computer circuit analysis programs for the design of integrated electronic circuits and systems. The examples chosen for this discussion are developed with program SPICE.[1] SPICE is a nodal analysis program which offers nonlinear dc, nonlinear transient and linear ac analysis of electronic circuits in a single program. Free format input, built-in device models (diodes and bipolar and field-effect transistors) and circuit nesting features make the program easy to use. The sparse matrix structures and the use of implicit integration and adjoint network techniques help to make SPICE an efficient simulation tool.

Fundamentals of the SPICE program structure will be discussed in the sub-sections which follow. The basic analysis approach, data structure considerations and a method for numerical interaction will be presented. Next, the method of adjoint network calculations will be developed and applications will be described. Finally, and most importantly, advanced MOS and bipolar transistor models will be

be developed as they are implemented in SPICE. The need for accurate models is of paramount importance and the requirements to correctly determine parameters for these models will become clear as one reads the project reports in subsequent sections.

COMPUTER-AIDED CIRCUIT ANALYSIS OVERVIEW

To initiate the discussion, the circuit example shown in Figure 1 will be used as a vehicle to illustrate the techniques of computer aided circuit analysis. The three input waveforms shown in Figure 1 drive the circuit so that:

a) the dc steady-state bias-point is achieved,

b) the small signal steady state behavior is determined (as a function of frequency), and

c) the transient output waveform is calculated.

Each of these analysis modes requires the solution of the nonlinear set of equations associated with the transistor. For cases a) and b) the solution with nonlinear elements must be found only once. For the small signal analysis (case b)) the subsequent frequency calculations use the model values linearized about the bias point. However, for case c), the transient analysis, the set of nonlinear equations must be solved at each time-point in the analysis. The solution of the nonlinear equations associated with the device model is thus the backbone of computer circuit simulation. Efficiency in achieving this solution is a major concern.

The nonlinear dc solution is achieved by iterative solution of the linearized equivalent circuits. The so-called "linearized equivalent circuit" is constructed using the first-order terms of the Taylor series expansion of the nonlinearities about some initial point. This particular iterative procedure - often called Newton-Raphson iteration[2] - solves for the unknown voltages in the Taylor-series expansion, updates the expansion about the newly found value and continues the process.*

---

* Reference (2) is included in Appendix I to provide further comments and references on techniques of Computer-Aided Circuit Analysis.

Each linearization is made using the solutions from the previous iteration. The iterations continue until the solution converges to within some specified tolerance of the value from the previous iteration - for example 10μV at every node. All energy storage elements are replaced by their dc equivalents for the dc solution. The network equations themselves (with linearized matrix entries for the model) are solved using a modified form of Gaussian elimination.[2] For SPICE, the equations are written based on Kirchhoff's Current Law for all the nodes in the circuit (i.e. nodal analysis). The simplest Gaussian elimination consists of constructing an upper triangular matrix from the original nodal admittance matrix, Y, and determining the node voltages, in sequence, starting from the bottom of the triangular matrix. This final solution method is called backward substitution. The SPICE implementation uses a matrix transformation to achieve two matrices U and L which are upper and low triangular with the property that LU = Y. The voltage vector is solved for by simple matrix operations involving L^{-1} and U^{-1}. The matrix operations are easy owing to the triangular structure of the matrices.

The procedures described above are the basis of essentially all nonlinear computer-aided circuit analysis. For small signal analysis, with frequency as a variable, the need for iterations is eliminated once the nonlinear dc solution is found. Specifically, the model is linearized about the dc operating point. The ac analysis involves arithmetic operations with complex arguments. However, the basic approach of the L-U transformation is used to solve for the complex voltage vector. The energy storage elements have been added to the Y matrix as complex admittances.

For transient analysis the procedure described for the nonlinear dc bias solution is used repetitively. Since energy storage elements are now included in the circuit, the solution of the integral-differential equations must be found.

The solution of the integral-differential equations proceed using numerical integration techniques.[2] For example, the branch relationship for an inductor, in integral form, is:

$$i_L(t+\Delta) = i_L(t) + \frac{1}{L}\int_t^{t+\Delta} v_L(\tau)\, d\tau$$

and the integral can be approximated as:

$$i_L(t+\Delta) = i_L(t) + \frac{\Delta}{L} v_L(t+\Delta)$$

The choice to approximate the integral using the yet to be determined voltage at $t+\Delta$ is a form of implicit integration. Now the branch relation for the inductor at time $t+\Delta$ has a form equivalent to a current source, $i_L(t)$, (a constant since we have just solved for its value) in parallel with a conductance of value Δ/L. A similar branch relation exists for capacitive elements. The result is that at each new time point $t+\Delta$, the energy storage elements are replaced with their equivalent circuits which approximate the integral-differential branch relationships. The nonlinear solution of the circuit equations can then proceed at that time point, just as originally described for the nonlinear dc condition.

Figure 1 and the above discussion have been used only to introduce the elements of computer-aided circuit analysis. The following subsections will expand these ideas and illustrate their implementation using the example in Figure 1 and the algorithms in program SPICE.

MODEL LINEARIZATION

Modeal linearization is the first step in obtaining the nonlinear dc bias point for an electronic circuit (for example the circuit shown in Figure 1). To show more precisely how the linearization proceeds, let us begin by considering the bipolar transistor model[3] showin fn Figure 2. The equations associated with the elements are:

$$I_{CE} = I_S \left(e^{qV_{BE}/kT} - 1\right) \tag{1}$$

$$I_{CC} = I_S \left(e^{qV_{BC}/kT} - 1\right) \tag{2}$$

$$C_E = C_{jE}(V_{BE}) + \frac{qI_{CE}}{kT} \cdot \tau_F \tag{3}$$

$$C_C = C_{jC}(V_{BC}) + \frac{qI_{CC}}{kT} \cdot \tau_R \tag{4}$$

where I_S is the base transport current ($I_S = \alpha_F I_{ES} = \alpha_R I_{CS}$ for conventional Ebers-Moll notation); V_{BE} and V_{BC} are the junction voltages; C_{jE} and C_{jC} are the voltage dependent depletion layer capacitances and τ_F and τ_R are the forward and reverse transit times which describe the diffusion capacitances due to minority charge storage.

The two diodes labeled as I_{CE/β_F} and I_{CC/β_R} in Figure 2 represent the forward and reverse contributions to base current, using the exponential relationships given by equations (1) and (2). The I_{CE} and I_{CC} terms are the forward and reverse base transport currents. The linearization of the four exponentially nonlinear elements shown in Figure 2 is essential to initiate the iterative solution for the dc bias point of the circuit shown in Figure 1. To illustrate the linearization procedure, consider the exponential diode relationship for I_{CE/β_F} as is shown in Figure 3a.

If an initial operating point V_{BEO} is chosen, a linearization about that point can be made by the parallel combination of the current source and conductance shown in Figure 3b. The element values are:

$$G_{PIFO} = \frac{\partial}{\partial V_{BE}} \left(\frac{I_{CE}}{\beta_F} \right) \Bigg|_{V_{BEO}} = \frac{qI_S}{kT\beta_F} \, e^{qV_{BEO}/kT} \tag{5}$$

$$I_{BEFO} = \left(\frac{I_{CE}}{\beta_F} \right) \Bigg|_{V_{BEO}} - G_{PIFO} V_{BEO} \tag{6}$$

In equation (6) the first term represents the actual base current for a V_{BEO} bias. Performing the same procedure for the other exponential relationships, the composite linearized dc model is shown in Figure 4. The "O" subscripts have been dropped since we will no longer restrict ourselves to the initial values of the linearization. For the linearization of I_{CE} and I_{CC} it should be noted that the conductances are in fact transconductance generator elements. The importance of this result will become apparent as the nodal admittance matrix is generated for the circuit shown in Figure 1. The additional terms in Figure 4 have values given by:

$$G_{PIR} = \frac{\partial}{\partial V_{BC}} \left(\frac{I_{CC}}{\beta_R} \right) \tag{7}$$

$$I_{BCR} = \frac{I_{CC}}{\beta_R} - G_{PIR} \cdot V_{BC} \tag{8}$$

$$G_{MF} = \frac{\partial}{\partial B_{VE}} \left(I_{CE} \right) \tag{9}$$

$$G_{MR} = \frac{\partial}{\partial V_{BC}} \left(I_{CC} \right) \tag{10}$$

The two current generators $\beta_F I_{BEF}$ and $\beta_R I_{BCR}$ could easily have been expressed in a form more like equation (6) by defining:

$$I_{CEF} \triangleq I_{CE} - G_{MF} \cdot V_{BE} \tag{11}$$

$$I_{CCR} \triangleq I_{CC} - G_{MR} \cdot V_{BC} \tag{12}$$

It follows immediately that $\beta_F I_{BEF} = I_{CEF}$ and $\beta_R I_{BCR} = I_{CCR}$, thus only the form given in Figure 4 is used.

The means for applying the results shown in Figure 4 can now be summarized. Before each new iteration to solve for the dc bias point of the circuit shown in Figure 1, the nonlinear equations describing the transistor model are linearized about the voltages from the previous iteration (or some default values for the first iteration). Figure 4 shows the linear elements used to represent the Bipolar Transistor. In the next subsection the entering of these model element values into the Y matrix will be demonstrated.

NODAL CIRCUIT ANALYSIS

A. MATRIX FORMULATION

Having considered the linearization of the bipolar junction transistor model (as an example of a nonlinear element to be utilized in a circuit design) we can now proceed to see how the linear elements are entered into a formulation which can be solved efficiently using numerical techniques. The approach used in SPICE is to construct and solve the Kirchhoff Current Law equations for each node of the circuit. The circuit shown in Figure 5 represents the bipolar circuit from Figure 1 with the transistor replaced with its linearized equivalent model shown in Figure 4. All energy storage elements have been removed from the circuit for the dc analysis. All node voltages are referenced to the ground node (which is numbered "0"). The current law equations for nodes 1–5 are written below:

$$
\begin{array}{l}
\text{Node 1:} \\
\text{Node 2:} \\
\text{Node 3:} \\
\text{Node 4:} \\
\text{Node 5:}
\end{array}
\begin{bmatrix}
(G_{IN}) & (-G_{IN}) & 0 & 0 & 0 \\
(-G_{IN}) & (G_{IN}+G_{PIR}+G_{PIF}) & (-G_{PIR}) & 0 & 0 \\
0 & (+G_{MF}-G_{MR}-G_{PIR}) & (G_C+G_{PIR}+G_{MR}) & (-G_C) & 0 \\
0 & 0 & (-G_C) & (G_C+G_L) & (-G_L) \\
0 & 0 & 0 & (-G_L) & (G_L)
\end{bmatrix}
\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \end{bmatrix}
= -
\begin{bmatrix}
I_{V_{IN}} \\
I_{BCR}+I_{BEF} \\
-I_{BCR}+\beta_F I_{BEF} \\
\quad -\beta_R I_{BCR} \\
0 \\
I_{V_{LL}}
\end{bmatrix}
\tag{13}
$$

The currents are taken as being positive out of each node. The terms I_{VIN} and I_{VLL} are the currents through the voltage sources. These current values are determined by other circuit elements and since the voltages are constant, the nodes can be rearranged so as to move them to the bottom of the matrix. The five equations were written directly from Figure 5. The following rules can be applied to write the equations knowing only the nodal connections of the elements. For conductance between nodes i and j the value is added to the diagonal entries y_{ii} and

y_{jj} and is subtracted from the off-diagonal entries y_{ij} and y_{ji}. For a transconductance the reference polarities and node numbering shown in Figure 6 can be considered. The transconductance term g_m is added to the matrix terms y_{ik} and $y_{j\ell}$ and is substracted from the $y_{i\ell}$ and y_{jk} terms. This usually results in a non-symmetric matrix structure. For the case shown in Figure 5, ℓ=J=0 for the G_{MF} generator, thus only the y_{32} term is non-zero. For the G_{MR} generator i=0 so that only the y_{33} and y_{32} entries are non-zero. It follows that equation (13) could have been written directly from a list of element node connections and prescribed linearized forms for the nonlinear device models. However, equation (13) is not necessarily in the most convenient order for efficient solution of the network equations. The voltages at nodes 1 and 5 are known, and it is convenient to reorder the matrix to utilize this fact. The Y matrices encountered in nodal circuit analysis generally contain many zero terms. These zero terms can easily account for 50-80% of the Y matrix. It is possible to take advantage of the sparsity of non-zero terms to reduce the memory required to store the Y matrix and to reduce the number of numerical operations required to solve the nodal equations. Appendix II contains a detailed description of one particular implementation[4] of what has come to be called "Sparse Matrix Techniques". The following section illustrates the features of the sparse matrix technique for efficient storage and L-U factorization. The circuit from Figure 5 is used as an example and the notation is that of Berry[4].

B. SPARSE MATRIX SOLUTION*

The first step is to reorder the rows and columns (as is shown in equation (14)) to: 1) move the known node voltages V_1 and V_5 to the bottom of the matrix; and 2) move V_3 to position shown in equation (14). The second step was needed to minimize fill-in during the L-U transformation. The details of checking for fill-in are discussed in the Appendix. The resulting equations are:

---

* While the sparse matrix approach can offer computational advantages it can also obscure the basic solution approach. Because of this conceptual draw-back it is suggested that the reader skip to equation (17) for the first pass.

$$\left[\begin{array}{ccc:cc} (G_{IN}+G_{PIR}+G_{PIF}) & 0 & (-G_{PIR}) & (-G_{IN}) & 0 \\ 0 & (G_C+G_L) & (-G_C) & 0 & (-G_L) \\ (+G_{MF}-G_{MR}-G_{PIR}) & (-G_C) & (G_C+G_{PIR}+G_{MR}) & 0 & 0 \\ \hdashline (-G_{IN}) & 0 & 0 & (G_{IN}) & 0 \\ 0 & (-G_L) & (0) & 0 & G_L \end{array}\right] \left[\begin{array}{c} V_2 \\ V_4 \\ V_3 \\ V_1 \\ V_5 \end{array}\right] = - \left[\begin{array}{c} (I_{BCR}+I_{BEF}) \\ 0 \\ (-I_{BCR}+\beta_F I_{BEF} \\ -\beta_R I_{BCR}) \\ I_{V_{IN}} \\ I_{V_{LL}} \end{array}\right] \qquad (14)$$

The 3 x 3 matrix shown in the upper left corner of equation (14) is the only portion of interest since V_1 and V_5 are known. What is done next is to create pointers to the non-zero entries in this 3 x 3 matrix. This is best illustrated by redrawing the matrix with 1 and 0 entries corresponding to non-zero and zero entries respectively. Corresponding to columns and rows 1-3 of equation (14) the following structure exists:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \left[\begin{array}{ccc} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{array}\right] \end{array} \qquad (15)$$

A column array is created which contains the diagonal Y matrix entries and the non-zero upper and lower triangular matrix entries in a prescribed sequence. Two arrays of pointers are created to identify the matrix indices of the non-zero terms stored in the column array. This is done in two passes: First pointers to every non-zero entry in the entire matrix are created, next this is collapsed to a set which points

either to upper or lower triangular entries. In the "first pass" array IUR has n entries corresponding to the n-nodes of the circuit. The i^{th} entry in IUR indicates the first position in the second array, IUC, which contains a non-zero element in the i^{th} row of the Y matrix. The IUC array contains the column positions of the non-zero off-diagonal entries in each row of the matrix in sequence. For the i^{th} row, IUR points to the first entry in IUC. Further entries in IUC contain subsequent column numbers of matrix positions. For the 3 x 3 matrix given in equation (15), the IUR and IUC arrays contain the following entries on the first pass:

IUR (1) = 1	IUC (1) = 3
IUR (2) = 2	IUC (2) = 3
IUR (3) = 3	IUC (3) = 1
	IUC (4) = 2

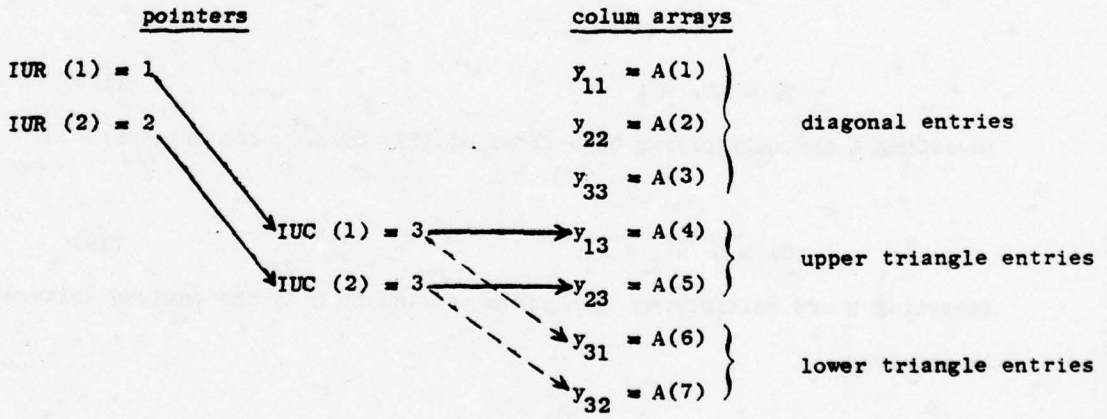
For example, IUR (3) "points" to IUC (3). The IUC (3) and IUC (4) entries indicate that columns 1 and 2 have non-zero terms in the 3^{rd} row. The "second pass" compares entries corresponding to the upper and lower triangular portions of the matrix. A set of pointers is then choosen so that the matrix is assumed symmetric. This new set of pointers correctly selects all non-zero entries but in addition may point to a limited number of zero terms corresponding to the non-symmetrical entries. For the example given by equation (15) the new pointers are:

IUR (1) = 1	IUC (1) = 3
IUR (2) = 2	IUC (2) = 3

Notice that only n-1 rows are now needed. Considering the upper portion of the matrix, IUR (2) points to IUC (2) which in turn indicates that the 3rd column of row two in upper triangle is non-zero. The same pointers refer to the lower triangle in that the IUR entries now go column by column and the IUC entries tell the row of non-zero entries. Thus IUR (2) indicates that in the 2nd column we must go to IUC (2) to find non-zero entries in the lower triangle of Y. IUC (2) tells us that the entry in the 3rd row of column two is non-zero. To see how the pointers and column array fit together, let us enumerate the Y matrix entries in equation (15).

$$\begin{matrix} y_{11} & 0 & y_{13} \\ 0 & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{matrix} \qquad (16)$$

The overall data structure is:



The diagonal entries are of known number and location. The index of IUC tells the total number of entries in either triangle. Thus the starting location of the upper triangle in the column array is n+1. For the first lower triangle entry, the

position is at n+1 incremented by the maximum index for IUC.

This rather long procedure has given us a sparse pointer structure which locates all non-zero entries in the Y matrix. Using these pointers, the following algorithm can be used to calculate the L and U matrices such that LU = Y. The procedure must be executed n times (where n is the number of circuit nodes) and the superscript indicates the i^{th} or $i\text{-}1^{st}$ row-column step in the procedure.

$$u_{ij} = y_{ij}^{(i-1)} \quad \text{for all } j \geq i$$

$$l_{ji} = \frac{y_{ji}^{(i-1)}}{u_{ii}} \quad \text{for } j > i$$

$$\text{and } y_{jk}^{(i)} = y_{jk}^{(i-1)} - l_{ji}\, u_{ik} \quad \text{for } j \text{ and } k \geq i$$

Once the entries in the L and U matrices are known, the solution for node voltages is straightforward. Namely,

$$Yv = LUv = i \tag{17}$$

Inverting L and multiplying both sides of (17) by L^{-1}, one obtains:

$$Uv = L^{-1}i \triangleq i^{*}. \tag{18}$$

Inverting U and multiplying both sides of (18) by U^{-1}, the desired voltage vector is obtained:

$$v = U^{-1}i^{*} = U^{-1}L^{-1}i \tag{19}$$

This procedure is used for nonlinear dc and transient analysis with iterative solutions as models are updated at each new set of node voltages. The procedure

is also appropriate for ac analysis using complex arithmetic and linearized model entries.

The circuit example given in Figure 5 is rather simple and does not give a fair picture of the storage and computation savings that are possible using the complete Berry algorithm. A more typical circuit to be analyzed and the resulting matrix structure are shown in Figures 7a) and b). For this case the number of zero-valued entries is substantial. The X's in Figure 7b) indicate the original Y matrix entries corresponding to the uncircled node numbers. The 0's indicate "fill-in" in the L and U matrices resulting from the decomposition. Figure 7c) shows the matrix structure for the same circuit but with the nodes renumbered as shown by the circled node numbers. It is clear from Figure 7c) that fill-in is considerably reduced as is shown by the substantial decrease in "0" entries. For the example cited in Figure 7, the following reductions in arithmetic operation counts can be achieved:

	DIV	MULT and ADDS
Straight LU decomposition	378	6930 ea
Renumbering and no sparse pointers	134	746 ea
Renumbering and sparse pointers	63	169 ea

The reordering aspect of the Berry algorithms is described in Appendix II.

The primary purpose of the discussion of sparse matrix techniques has been to illustrate the creation of the pointer structure needed for sparse storage and the steps needed for the L-U factorization. A general result which can be stated regarding the computational efficiency of using Sparse Matrix Structures is that the long operations count (multiplications for example) descreses from $n^3/3$ to a number more nearly proportional to n, where n is the number of nodes in the circuit.

NUMERICAL INTEGRATION

As was suggested in the overview, numerical integration approximations are used to solve the integral-differential circuit equations in the time domain. The form of the integration rule which is used can make a significant difference in both the speed and accuracy of the computer-simulated results. If we consider the integration of a function f(t), three common approximations[2] which can be made are:

$$\int_{t}^{t+\Delta} f(\tau)d\tau \approx \begin{cases} \Delta \cdot f(t) \\ \Delta \cdot f(t+\Delta) \\ \frac{\Delta}{2}\left[f(t) + f(t+\Delta)\right] \end{cases} \qquad (20)$$

These integration rules are called Forward-Euler (F-E), Backward-Euler (B-E) and Trapezoidal (TR) respectively. The first method is an explicit scheme in that it uses a known function value to approximate the integral. The Backward-Euler and Trapezoidal are implicit in that they use a not-yet-determined value of the function to approximate the integral. The use of the implicit integration rules has been an important advance for computer circuit simulation owing to the improved stability of these methods over Forward-Euler for a given step size. It is well known[2] that the explicit integration may proceed no faster than the smallest time constant of the circuit to maintain a stable solution. In the discussion presented here, the accuracy and stability properties of B-E and TR integration formulae will be developed and compared with that for the F-E method.

First, let us understand exactly how the integration formulae are applied to computer-aided circuit analysis. It has previously been shown that for an inductor the branch relationship requires that:

$$i_L(t+\Delta) = i_L(t) + \frac{1}{L}\int_t^{t+\Delta} v_L(\tau)d\tau$$

Hence our numerical approximations to the integral can be applied directly. The results are:

$$i_L(t+\Delta) = \begin{cases} i_L(t) + \frac{\Delta}{L} v_L(t) & \text{(F-E)} \\ i_L(t) + \frac{\Delta}{L} v_L(t+\Delta) & \text{(B-E)} \\ i_L(t) + \frac{\Delta}{2L} v_L(t) + \frac{\Delta}{2L} v_L(t+\Delta) & \text{(TR)} \end{cases} \qquad (21)$$

These three results can be represented as the branch equivalent-circuits shown in Figure 8a. These element-values are entered in the Y matrix at each time point and the solutions for $t+\Delta$ is found using Newton iterations. *Note that* the current generator values may change as time proceeds. If the program includes variable time-increments, Δ, then the conductances also change value with time.

Constant-valued capacitive elements impose the following branch relationship:

$$i_C(t) = C\frac{dv_C}{dt}(t) \qquad (22)$$

which can be rewritten as:

$$\frac{1}{C}\int_t^{t+\Delta} i_C(\tau)d\tau = v_C(t+\Delta) - v_C(t) \qquad (23)$$

Again we can apply the integration rules to obtain the relationships:

$$v_C(t+\Delta) = \begin{cases} v_C(t) + \frac{\Delta}{C}\, i_C(t) & \text{(F-E)} \\ v_C(t) + \frac{\Delta}{C}\, i_C(t+\Delta) & \text{(B-E)} \\ v_C(t) + \frac{\Delta}{2C}\, i_C(t) + \frac{\Delta}{2C}\, i_C(t+\Delta) & \text{(TR)} \end{cases} \qquad (24)$$

These results can be represented as the branch equivalent circuits shown in Figure 8b. The circuits shown in parentheses are the Norton equivalents which are more appropriate for nodal analysis.

Thus far, only constant valued energy storage elements have been considered. Assume that the branch relationship is now defined by the equation:

$$i_C(t) = \frac{\partial}{\partial t}\, Q\left(v_C(t)\right) \qquad (25)$$

Integrating both sides of the equation yields

$$\int_t^{t+\Delta} i_C(\tau)\, d\tau = Q\left(v_C(t+\Delta)\right) - Q\left(v_C(t)\right) \qquad (26)$$

For this case the integral must be approximated, as before, but in addition the nonlinear Q relationships must be linearized. The result is that Newton-Raphson iterative procedure must now be applied to nonlinear energy storage elements as well as to the dc transistor model nonlinearization.

A primary result of the preceeding discussion is that equivalent circuits are created to approximate the integral-differential equations associated with energy storage elements. These equivalent circuits are used with the standard iterative matrix solution methods developed for nonlinear dc analysis to solve for

circuit behavior in the time domain. The question of relative accuracy and stability of the several integration methods (F-E, B-E and TR) is now discussed in the context of the simple circuit example shown in Figure 9. The voltage v_R is the variable of interest. It is apparent from the circuit that

$$v_R(t) = E - v_C(t) \tag{27}$$

so that

$$v_R(t+\Delta) = E - v_C(t+\Delta) \tag{28}$$

We can now apply the integration formulae given by equation (24) to approximate $v_C(t+\Delta)$. That is, using equation (24) in equation (28):

$$v_R(t+\Delta) = E - v_C(t) - \begin{cases} \frac{\Delta}{C}\, i_C(t) & \text{(F-E)} \\ \frac{\Delta}{C}\, i_C(t+\Delta) & \text{(B-E)} \\ \frac{\Delta}{2C}\, i_C(t) + \frac{\Delta}{2C}\, i_C(t+\Delta) & \text{(TR)} \end{cases} \tag{29}$$

It is true that for any time t (or t+Δ):

$$i_C(t) = i_R(t) = \frac{V_R(t)}{R} \tag{30}$$

Thus using equations (27) and (30) in equation (29) one obtains:

$$v_R(t+\Delta) = v_R(t) - \begin{cases} \frac{\Delta}{RC}\, v_R(t) & \text{(F-E)} \\ \frac{\Delta}{RC}\, v_R(t+\Delta) & \text{(B-E)} \\ \frac{\Delta}{2RC}\, v_R(t) + \frac{\Delta}{2RC}\, v_R(t+\Delta) & \text{(TR)} \end{cases} \tag{31}$$

These equations can be rearranged to give the following recursive relationships:

$$v_R(t+\Delta) = \begin{cases} v_R(t)\left(1 - \frac{\Delta}{\tau}\right) & \text{(F-E)} \\ v_R(t)\left(\frac{1}{1 + \frac{\Delta}{\tau}}\right) & \text{(B-E)} \\ v_R(t)\left(\frac{1 - \frac{\Delta}{2\tau}}{1 + \frac{\Delta}{2\tau}}\right) & \text{(TR)} \end{cases} \qquad (32)$$

where $\tau = RC$. For a step voltage input, the initial value of v_R is $v_R(o) = E_o$. By applying the formulae given by equation (32) repetitively for n equal Δ's the result is:

$$v_R(n\Delta) = \begin{cases} E_o\left(1 - \frac{\Delta}{\tau}\right)^n & \text{(F-E)} \\ E_o\left(\frac{1}{1 + \frac{\Delta}{\tau}}\right)^n & \text{(B-E)} \\ E_o\left(\frac{1 - \frac{\Delta}{2\tau}}{1 + \frac{\Delta}{2\tau}}\right)^n & \text{(TR)} \end{cases} \qquad (33)$$

The exact solution for this circuit and step stimulation is:

$$v_R(t) = E_o e^{-\frac{t}{\tau}} \qquad \text{(EXACT)} \qquad (34)$$

so that for $t = n\Delta$:

$$v_R(n\Delta) = E_o e^{-\frac{n\Delta}{\tau}} \qquad \text{(EXACT)} \qquad (35)$$

Several comparisons can now be made using equations (33) and (35). For a single step (n=1) the accuracy can be compared for several values of Δ. The results for $\Delta = .1\tau$ and $\Delta = \tau$ are given below.

ACCURACY:

	F.E. $\left(1 - \frac{\Delta}{\tau}\right)$	B.E. $\frac{1}{1 + \frac{\Delta}{\tau}}$	TR. $\frac{1 - \frac{\Delta}{2\tau}}{1 + \frac{\Delta}{2\tau}}$	EXACT $e^{-\frac{\Delta}{\tau}}$
$\Delta = 0.1\tau$	0.90000	0.90909	0.90476	0.90483
$\Delta = \tau$	0.0	0.5	0.333	0.368

Clearly it can be concluded that the trapezoidal approximation gives the most accurate approximation to the exact solution for a given time step Δ. The stability of the methods can be demonstrated in an approximate sense by seeing for what values Δ the solution begins to oscillate with odd and even values of n. The results given below show several things.

STABILITY:

	F.E. $\left(1 - \frac{\Delta}{\tau}\right)^n$	B.E. $\frac{1}{\left(1 + \frac{\Delta}{\tau}\right)^n}$	TR. $\left(\frac{1 - \frac{\Delta}{2\tau}}{1 + \frac{\Delta}{2\tau}}\right)^n$
STABLE FOR	$0<\Delta<\tau$	$0<\Delta$	$0<\Delta<2\tau$
FOR VERY LARGE Δ $\frac{\Delta}{\tau} \gg 1$	$\left(-\frac{\Delta}{\tau}\right)^n$ DIVERGENT AND OSC.	$\frac{1}{\left(\frac{\Delta}{\tau}\right)^n}$ STABLE	$(-1)^n$ STABLE BUT OSC.

First, the solutions exhibit oscillations only for F-E and TR when Δ is greater than τ and 2τ respectively. The B-E formulation is <u>stable</u> in the sense that the $\left(\frac{1}{I + \frac{\Delta}{\tau}}\right)^n$

recursive expression never oscillates in sign. The second comment to be made is with regard to stability for very large values of Δ. This consideration shows that F-E not only oscillates but diverges from the correct answer. Both B-E and TR are stable. However, for TR the oscillations still persist for conditions such that $\Delta > 2\tau$.

Program SPICE uses the trapezoidal integration rule. Thus the solutions are of the greatest accuracy for the correct choice of time increments. However, for increments which are too large ringing can be observed in the waveforms. This is most often eliminated by decreasing the internal program integration increments while maintaining the same output plot increments. It might be thought that the backward euler formulation would be a better approach owing to its unconditional stability. However, one major drawback should be stated. There is no easy way of telling when accuracy is degraded due to excessive time-step increments. With trapezoidal, the oscillation problems occur at about the step increment where accuracy is also degraded. Thus, by using a proper increment step control both the accuracy and stability can be maintained for the trapezoidal method.

ADJOINT NETWORK ANALYSIS

In the preceeding subsections numerical and network analysis techniques have been described which are the basis of computer-aided circuit analysis. In this section the principles and application of the adjoint network will be described. Although the topic may at first sound theoretical and unrelated to practical circuit analysis, in reality the technique is easy to apply and has had a major impact on integrated circuit design using the computer. Specifically, adjoint network techniques are used to provide efficient noise analysis,[5] dc and ac sensitivities of circuit outputs to parameter variations[6] and finally "design optimization". One hesitates to use this last term owing to the many misconceptions as to its meaning. In fact, the adjoint network can be used along with the standard network techniques to perform two types of analysis which together might be called design optimization. Sensitivity analysis is used to determine the gradient of a circuit output with respect to the circuit parameters. A performance index can be defined and the deviation from this index can be minimized using the gradient information and some algorith to seek the minimum. For example, the Fletcher-Powell search algorithm is commonly used.[7] This approach is called design optimization. Stated more exactly, one can say that sensitivity analysis is used iteratively with some search algorithm to minimize an error function. A minimum error should thus guarantee an "optimum" design. Unfortunately, circuit parameter tolerances about this optimum design can cause a poor production yield. This is especially true for integrated circuits where exact component values are difficult to guarantee. For IC's it may be more advantageous to define an optimum design in terms of a production yield. Adjoint network analysis techniques can be used to perform worst-case and statistical analysis. In both cases the adjoint network provides an efficient means to analyze the

effects of numerous possible circuit variations The added information from such simulations can lead to an optimum component value and tolerance assignment. In this case the objective might be to optmize yield. One can see that the efficient analysis of many possible component values is the key to optimizing a design.* Use of the adjoint network provides these needed analyses with a minimum increase in computational expense.

To compute the sensitivities for a variable, say the changes in voltage ΔV_I across a current source I_T, we assume that an expression for V_I can be written:

$$V_I = f(p_1, p_2 \ldots p_p) \tag{36}$$

where the p's represent the parameters which give rise to the observed ΔV_I. The change, ΔV_I is then given by:

$$\Delta V_I = \sum_{k=1}^{p} \frac{\partial f}{\partial p_k} \cdot \Delta p_k \tag{37}$$

and each term $\frac{\partial f}{\partial p_k}$ represents the sensitivity of V_I to the change in parameter p_k. Our task then is to find an expression of the form of equation (37) and to identify the proper sensitivity terms. Toward this end the adjoint network is now introduced. This mathematical formulation along with the proper reordering and identification of terms will facilitate the desired result.

The starting point, for considering the adjoint network is Tellegen's Theorem.[8] The theorem states that if we have two topologically identical network (i.e. the same graph structure, branch numbers, node numbers and orientations) - call them η and $\hat{\eta}$ with V_k-I_k and Ψ_k-Φ_k voltage-current

---

* It should also be realized, that the real key is to have a good design to begin with. However even initial designs can be marked improved by knowing sensitivity information.

identifications respectively - then

$$\left.\begin{aligned} \sum_{k=1}^{b} V_k \Phi_k &\equiv 0 \\ \sum_{k=1}^{b} \Psi_k I_k &\equiv 0 \end{aligned}\right\} \qquad (38)$$

where b is the total number of branches of the network. Figure 10 shows the networks η and $\hat{\eta}$ schematically, along with a network $\eta+\Delta\eta$ which represents incremental pertubations to the voltage and current vectors of network η. All three networks satisfy Tellegen's theorem and it can be easily shown by application of equation (38) that $\Delta\eta$ also satisfies the theorem so that:

$$\left.\begin{aligned} \sum_{k=1}^{b} \Delta V_k \Phi_k &\equiv 0 \\ \sum_{k=1}^{b} \Psi_k \Delta I_k &\equiv 0 \end{aligned}\right\} \qquad (39)$$

Subtracting the above equations from one another it follows that:

$$\sum_{k=1}^{b} \left[\Delta V_k \Phi_k - \Psi_k \Delta I_k\right] \equiv 0 \qquad (40)$$

This equation is fundamental to determination of sensitivities and the definition of the adjoint network.

Consider now the possible branch relationships defined by Figure 11 for current sources, voltage sources, resistors and conductances. Equation (40) can be rewritten in terms of this notation with summations taken over the respective number of branches for each element type.

$$\sum_{V}\left[\Delta V_v\Phi_v - \Psi_v\Delta I_v\right] + \sum_{I}\left[\Delta V_I\Phi_I - \Psi_I\Delta I_I\right] \tag{41}$$

$$+\sum_{R}\left[\Delta V_R\Phi_R - \Psi_R\Delta I_R\right] + \sum_{G}\left[\Delta V_G\Phi_G - \Psi_G\Delta I_G\right] \equiv 0$$

The exact relationship for the change in voltage across a resistor, for example, is:

$$V_R+\Delta V_R = \left(R_R+\Delta R_R\right)\left(I_R+\Delta I_R\right) \tag{42}$$

If the second order term in $\Delta R_R\Delta I_R$ is neglected then it follows that:

$$\Delta V_R \approx I_R\ \Delta R_R+R_R\Delta I_R \tag{43}$$

using equation (43) in the summation over resistive branches in equation (41) the result is:

$$\sum_{R}\left(\Delta V_R\Phi_R - \Psi_R\Delta I_R\right) \approx \sum_{R}\left[\Delta R_R I_R\Phi_R + R_R\Delta I_R\Phi_R - \Psi_R\Delta I_R\right] \tag{44}$$

$$\approx \sum_{R}\left[\underbrace{\Delta R_R I_R\Phi_R}_{(a)} + \underbrace{\Delta I_R}_{(b)}\underbrace{\left(R_R\Phi_R - \Psi_R\right)}_{(c)}\right]$$

The three terms identified as (a), (b) and (c) are now considered. We want to determine variations based on (a), however term (b) also enters into the summation and is an unknown quantity. By choosing the relationship in (c) so that:

$$\Psi_R \triangleq R_R\Phi_R \tag{45}$$

then the second term in equation (44) goes to zero and only the desired term is left. What has been done is to define the <u>adjoint</u> branch relationship given by equation (45) so that this happens. Notice that this so-called adjoint

network $\hat{\eta}$ satisfies Tellegen's theorem with regard to η but the branch relationships may be different. The same result is obtained for conductances so that:

$$\sum_G \left(\Delta V_G \Phi_G - \Psi_G \Delta I_G\right) = \sum_G -\Delta G_G V_G \Psi_G \tag{46}$$

and

$$\Phi_G \triangleq G_G \Psi_G \tag{47}$$

It can be seen from equations (45) and (47) that resistances and conductances translate into the same relationships in the adjoint network. Using equations (44) and (46) in equation (41):

$$\sum_V \left[\Delta V_V \Phi_V - \Psi_V \Delta I_V\right] + \sum_I \left[\Delta V_I \Phi_I - \Psi_I \Delta I_I\right]$$

$$+ \sum_R \Delta R_R I_R \Phi_R + \sum_G -\Delta G_G V_G \Psi_G = 0$$

which can be rearranged so that:

$$\sum_V \Psi_V \Delta I_V + \sum_I -\Delta V_I \Phi_I =$$

$$\sum_V \Delta V_V \Phi_V + \sum_I -\Delta I_I \Psi_I + \sum_R \Delta R_R I_R \Phi_R + \sum_G -\Delta G_G V_G \Psi_G \tag{48}$$

It should be noted that all summations on the right-hand side of equation (48) contain variations in the element values themselves (i.e. ΔV_V, ΔI_I, ΔR_R and

ΔG_G) while the variations on the left-hand side of the equation pertain to either voltages across current sources or currents through voltage soucres (ΔV_I and ΔI_V respectively).

It is now a fairly straightforward matter to show how equation (48) can be reduced to the desired result given by equation (37). We have defined branch relationships in the adjoint network. By an appropriate choice of stimulations of the adjoint, Ψ_V and Φ_I, the respective Φ_V, Ψ_I, Φ_R and Ψ_G can be solved for. If we choose all Ψ_V and Φ_I to be zero *except for the* single current source for which we want to determine output voltage sensitivity we obtain:

$$-\Delta V_{I_n} \Phi_{I_n} \Big|_{\substack{\text{all other } \Psi_V \text{ and } \Phi_I \\ \text{are zero}}} = \sum_V \Delta V_V \Phi_V + \sum_I \Delta I_I \Psi_I + \sum_R \Delta R_R I_R \Phi_R + \sum -\Delta G_G V_G \Psi_G \qquad (49)$$

By setting $\Phi_{I_n} = -1$ the form of equation (49) looks very much like that of equation (37). In fact, the sensitivity of V_{I_n} to changes in R_{R_k} for example is then $\frac{\partial f}{\partial R_{R_k}} = I_{R_k} \Phi_{R_k}$. To determine these sensitivities one appropriately stimulates the adjoint network and solves for the Ψ and Φ vectors. Since the original V and I vectors are known, the sensitivities can be constructed.

Several factors must still be discussed before the adjoint network method is in a form suitable for application in computer aided circuit analysis. Branch relationships for all elements (i.e. storage elements and controlled sources) must be defined. In addition, the approach must allow calculation of sensitivities across elements other than the restricted case for voltage and current sources which is suggested by equation (48). Finally, a means

for the implementation of the method must be described.

The definition of all elements in the adjoint is shown symbolically in Figure 12. For purposes of nodal circuit analysis the transconductance element is the single most important relationship for active circuits which was not considered in equation (48). For a voltage-controlled current source with input V_1 and $I_1 = 0$ and output $I_2 = gmV_1$ with V_2 arbitrary, the adjoint relationships are:

$$\begin{aligned} \Phi_2 &= 0 \\ \Phi_1 &= gm\Psi_2 \\ \Psi_1 &, \text{ arbitrary} \end{aligned} \tag{50}$$

These relationships are shown in Figure 12.

The application of the adjoint method to calculate variations across elements other than sources is a straightforward matter. If the current through some arbitrary element is desired, then a zero-valued voltage source is entered in series into the network and the ΔI_V can be calculated as described above. Similarly voltages across elements can be calculated by adding zero-valued current sources in parallel with the elements for which ΔV_I is desired. The net result is that the original circuit is modified appropriately so that the desired output ports can be stimulated in the adjoint network to determine sensitivities.

The method of computer implementation for the adjoint is found to be straightforward. To compute Ψ and Φ, the adjoint matrix $\hat{Y}$ is needed. However since η and $\hat{\eta}$ are interreciprocal:

$$\hat{Y} = Y^T \tag{51}$$

thus:

$$Y^T\Psi = \Phi \tag{52}$$

and using the L-U factorization:

$$U^T L^T \Psi = \Phi \tag{53}$$

It should be clear that once L and U have been constructed for the original network, the subsequent adjoint analysis to solve for Ψ and Φ is considerably less time consuming. In fact to a good approximation, if the original solution for V takes a unit of time, the subsequent solution for Ψ take .1 units.

In conclusion, the adjoint network is an efficient means to compute circuit sensitivities. The adjoint network results from an application of Tellegen's theorem and the proper definition of branch relationships in that network. The adjoint admittance matrix is obtained simply since:

$$\hat{Y} = Y^T = U^T L^T \tag{54}$$

The operations to obtain U^T and L^T are trivial once U and L are known. Thus the adjoint calculations require a minimal amount of additional analysis time. The sensitivity calculations are the basis for computer circuit optimization using gradient search methods. Sensitivities can also be used to perform noise, statistical and worst-case analysis.

BIPOLAR AND MOS TRANSISTOR MODELING

The linearization of a simple bipolar transistor model[3] was used earlier to demonstrate model implementation for computer-aided circuit analysis. In this subsection the specific features of the Gummel-Poon[9] bipolar transistor model and the Shichman-Hodges[10] MOS transistor model will be discussed. However, to understand the need for such advanced model formulations consider again the simple bipolar transistor model shown in Figure 2. The collector current characteristics for this model are shown in Figure 13a. The additional characteristics in the figure show the assumed voltage dependences for I_B and I_C and the resulting current gain dependence on I_C. Finally the transit time is assumed constant with current level. The junction space-charge capacitances have the familiar voltage dependence:

$$C(V) = \frac{C_{JO}}{\left(1-\frac{V}{\Phi}\right)^m} \qquad (55)$$

where C_{JO} is the zero-bias capacitance, Φ is the junction potential, m is a coefficient ranging typically from 1/2 to 1/3 and V is the junction voltage (positive for forward bias). When the experimental behavior of integrated bipolar transistors is observed, the characteristics are more nearly those depicted in Figure 13b. Namely, the I_C vs V_{CE} behavior shows a non-zero slope in the normal-active region which is the device output conductance resulting from base-width modulation. In addition, current gain is not constant with collector current which results from the low-level recombination and high-level injection effects shown by the I_C and I_B curves as a function of V_{BE}. Finally the high current effects also increase the transit time resulting in a decreased cut-off frequency. The measured junction space-charge capacitance can still be adequately modeled by equation (55). To achieve model behavior for the bipolar transistor which correctly predicts

the characteristics shown in Figure 13b, second order device effects must be accounted for.

A. THE BIPOLAR TRANSISTOR

The Gummel-Poon model incorporates the second order effects described above within the basic model formulation shown in Figure 2. The transport current is modified in the Gummel-Poon model by dividing I_S by the normalized based charge term, Q_B. Q_B is the total majority charge stored in the base region. Base width modulation tends to decrease Q_B and the collector current thus increases. High level injection adds majority charge to the base, thus increasing Q_B and decreasing I_C. The implementation of the Gummel-Poon model in SPICE defines Q_B in terms of the following parameters:

$$Q_1 = 1 + \frac{V_{BC}}{V_A} + \frac{V_{BE}}{V_B} \tag{56}$$

$$Q_2 = \frac{I_S}{I_k}\left[\exp\left(\frac{qV_{BE}}{kT}\right)-1\right] + \frac{I_S}{I_{kr}}\left[\exp\left(\frac{qV_{BC}}{kT}\right)-1\right] \tag{57}$$

$$Q_3 = \frac{1}{2}\left[Q_1 + \sqrt{Q_1^2 + 4Q_2}\right] \tag{58}$$

where I_S, I_k and V_A are defined in Figure 13b for the normal-active case. The parameters I_{kr} and V_B are similarly defined from the reverse-active characteristics. The I_S intercept is the same for forward and reverse operation by the nature of the transport model formulation.[9] The Q_2 term represents majority carrier base charge that is added via injection. The Q_1 term represents space-charge widening effects on Q_B. Equation (56) assumes that changes in base charge are linear with junction voltage thereby implying constant junction capacitances.

The resulting expression for I_{CE}-I_{CC} (see Figure 2) is:

$$I_{CE}-I_{CC} = \frac{I_S}{Q_B}\left[\exp\left(\frac{qV_{BE}}{kT}\right) - \exp\left(\frac{qV_{BC}}{kT}\right)\right] \tag{59}$$

Base current in the SPICE Gummel-Poon model consists of four terms. The diode terms shown in Figure 2 represent the ideal components. That is, these base current terms have an $\exp\left(\frac{qV}{kT}\right)$ dependence and give curves parallel to the I_C and I_E curves for forward and reverse operation respectively. The ideal components are described in terms of β_{FM} and β_{RM}, the maximum forward and reverse current gains. The non-ideal components represent space-charge, surface and other recombination effects. Their voltage dependence is of the form $\exp\left(\frac{qV}{nkT}\right)$ where n_e is the parameter for the base-emitter junction and n_c is that for the base-collector junction. The transport coefficients are given in terms of C_2I_S and C_4I_S for the normal and reverse terms respectively. Figure 13b shows the graphical interpretation of n_e and C_2I_S for the normal-active operation mode. The total expression for I_B is:

$$I_B = \underbrace{\frac{I_S}{\beta_{FM}}\left[\exp\left(\frac{qV_{BE}}{kT}\right)-1\right] + C_2I_S\left[\exp\left(\frac{qV_{BE}}{n_e kT}\right)-1\right]}_{(a)}$$

$$\underbrace{+\frac{I_S}{\beta_{RM}}\left[\exp\left(\frac{qV_{BC}}{kT}\right)-1\right] + C_4I_S\left[\exp\left(\frac{qV_{BC}}{n_c kT}\right)-1\right]}_{(b)} \qquad (60)$$

The term (a) in equation (60) replaces the I_{CE}/β_F relationship in Figure 2 and the (b) term replaces I_{CC}/β_R.

The end result is that nine parameters — I_S, V_A, V_B, C_2, I_k, n_e, C_4, I_{kr} and n_c — model the dc effects of base-width modulation, recombination and high-level injection effects in bipolar transistors. Changes in base transit time owing to high-level injection are incorporated via the expression:

$$\tau_f(\text{effective}) = \tau_F Q_B \qquad (61)$$

The complete set of 24 Gummel-Poon parameters is given in the SPICE user's guide in Appendix III. Of the 24 parameters, 15 are the same as those needed for the basic Ebers-Moll model. Thus only an additional nine must be determined to completely specify the Gummel-Poon model.

B. THE MOS TRANSISTOR

The modeling of MOS transistors for computer implementation is considerably easier than is bipolar transistor modeling. The incorporation of recombination and high-level injection effects which were necessary for bipolar devices is unnecessary for MOS transistors. However, the effect of bulk charge on channel conduction and channel length (charge) modulation due to drain voltage are important effects which must be included in second-order computer models. Additional effects, which can be significant, include bias dependent mobility and gate capacitance. However, these parameters are taken as constant in the Shichman-Hodges MOS model which is implemented in SPICE. The SPICE model does include voltage dependent drain and source junction space-charge capacitances. A 1/2-power-law voltage dependence is assumed.

The drain current for the MOS is described accurately by the equation:

$$I_D = \underbrace{\frac{Z\mu C_{ox}}{L}}_{\Delta\beta} \left\{ \left(V_{GS} - V_{FB} - 2\phi_F - \frac{V_{DS}}{2} \right) \cdot V_{DS} - \frac{2}{3} \underbrace{\frac{\sqrt{2\epsilon_s \epsilon_{ox} N_{sub}}}{C_{ox}}}_{\Delta\gamma} \left[\left(V_{DB} + 2\phi_F \right)^{3/2} - \left(V_{SB} + 2\phi_F \right)^{3/2} \right] \right\} \qquad (62)$$

where:

G,S,D,B - are the four terminal subscripts and voltages are positive-negative as indicated by the subscripts.

C_{ox} - is gate oxide capacitance per unit area

N_{sub} - is the substrate doping

$2\phi_F$ - is the strong inversion potential and is equal to $2kT/q \ln\left(N_{sub}/n_i\right)$

V_{FB} - is the flat-band voltage given as $\phi_{MS} - \frac{Q_{SS}}{C_{ox}}$ where:

ϕ_{MS} - is the metal-semiconductor work function

Q_{SS} - is the net surface charge.

The other terms have their standard meanings. The so-called "bulk charge" contribution to channel conduction is given by the second term in equation (62) with its 3/2 - power dependence. The gain factor β can be modified to include channel length modulation as follows:

$$\beta = \underbrace{\frac{Z\mu C_{ox}}{L}}_{\Delta\beta_o} \left[\frac{1}{1-\frac{\Delta L(V)}{L}}\right] \qquad (63)$$

The voltage dependence of $\Delta L(V)$ determines the appropriate channel length variations. It should be commented that mobility variations with bias can easily be included in equation (63) by using a variable μ.

The SPICE Shichman-Hodges MOS model uses simplified equations to model the general effects described by equations (62) and (63). To model the "bulk charge" contribution to conductance the threshold voltage is modified by a voltage dependent term. In essence the bulk charge effect is taken as a constant during integration along the channel so that a 3/2 power dependence is not obtained. That is:

$$V_T = V_{FB}+2\phi_F+\frac{Q_B}{C_{ox}} = V_{FB}+2\phi_F+\underbrace{\frac{\sqrt{2\varepsilon_S\varepsilon_{OX}N_{SUB}}}{C_{ox}}}_{\Delta\gamma}\left(V_{SB}+2\phi_F\right)^{\frac{1}{2}} \qquad (64)$$

If V_{TO} is defined to be the zero-V_{SB} value of V_T, which would be the measured threshold for no "back-gate" bias, then:

$$V_T = V_{TO} + \gamma\left[\left(V_{SB}+2\phi_F\right)^{\frac{1}{2}} - \left(2\phi_F\right)^{\frac{1}{2}}\right] \tag{65}$$

the quantities V_{TO}, γ and $2\phi_F$ are thus input parameters. This is a mixture of measured and calculated parameters since V_{TO} can be measured directly, γ can be determined from a series of "back-gate" bias experiments and $2\phi_F$ is calculated.

The Shichman-Hodges model assumes that output conductance, $G_{DSAT} = \partial I_D/\partial V_{DS}\,|_{V_{DS}>V_{DSAT}}$, is proportional to drain current in saturation (I_{DSAT}). That is:

$$G_{DSAT} = \lambda I_{DSAT} \tag{66}$$

where lambda is a channel length modulation parameter. The model implementation incorporates the effect into the current expression. If we define I_{DO} as*:

$$I_{DO} = \frac{\beta o}{2}\Big[\underbrace{\left(V_{GS} - V_T\right)^2}_{\triangleq F_1} - \underbrace{\left(V_{GD} - V_T\right)^2}_{\triangleq F_2}\Big] \tag{67}$$

the drain current expression which corresponds to the appropriate output conductance given by equation (66) is then:

$$I_D = I_{DO}\left(1 + \lambda V_{DS}\right) \tag{68}$$

---

* as written, this formulation applies only above threshold and for neither end of the channel pinched-off. For normal mode saturation $F_2 = 0$, and for reverse mode saturation $F_1 = 0$.

The $\left(1 + \lambda V_{DS}\right)$ term given in equation (68) can be interpreted as the equivalent to the $\left[\frac{1}{1 - \frac{\Delta L(V)}{L}}\right]$ term in equation (63). If $\Delta L(V)$ is assumed proportional to V_{DS} then:

$$\left[\frac{1}{1 - \lambda V_{DS}}\right] \approx 1 + \lambda V_{DS} \text{ for } \lambda V_{DS} \ll 1 \qquad (69)$$

where λ is the channel length modulation parameter. In essence the approximation for changes in L with V_{DS} is equivalent to the linear charge relationship assumed by using V_A for the Gummel-Poon model.

In summary, the MOS model built into SPICE requires 14 parameters for its complete specification. The dc parameters described by equations (63), (64), (65) and (66) are βo, γ, V_{TO}, $2\emptyset_F$ and λ. For the charge storage elements the oxide-capacitances associated with gate-source, gate-drain and gate-bulk are assumed to be constant. Junction capacitances of the source-bulk and drain-bulk have a 1/2-power dependence with user specified zero-bias values. The bulk junction potential and saturation current, as well as source and drain series resistance are supplied by the user. The complete set of parameters for the Shichman-Hodges model is listed in the SPICE user's guide in Appendix III.

CONCLUSION

This section of the report has presented techniques and model formulations which are basic to a sophisticated computer circuit analysis program. The examples have been taken from program SPICE which is being used at Stanford for a number of integrated circuit and system design problems. In the sections which follow, reports on specific modeling and circuit design problems will be presented. The emphasis on ac and transient analysis capabilities is about equal. It will also become apparent from the reports that modeling and model parameter determination are major factors in determining simulation accuracy and, in some cases, credibility.

APPLICATIONS REPORT TITLES

Project	Investigator
Characterization and Model Parameter Determinations for Program SPICE	H. E. Mussman
Modeling and Application of Uniform Distributed RC Devices	L. Gerzberg
Lumped Model Assessment	N. Chan
Analysis of a 6 MHz Oscillator Circuit for Ultrasonic Applications	H. V. Allen
Computer-Aided Design of Micropower Operational Amplifiers	S. R. Combs
High Voltage D-MOS Level Shifting Circuits	M. D. Pocha
Feasibility and Limitations Study of a Low Power, High Sensitivity Photo Detector	T. R. Gheewala

REFERENCES:

1. L. W. Nagel, D. O. Pederson

"Simulation Program with Integrated Circuit Emphasis"

16th Midwest Symposium on Circuit Theory, April 12, 1973, Waterloo, Ontario.

2. W. J. McCalla, D. O. Pederson

"Elements of Computer-Aided Circuit Analysis"

IEEE Trans. Circuit Theory, CT-18, p. 14-26, January 1971.

3. W. J. McCalla, W. G. Howard

"BIAS-3 - A Program for the Nonlinear DC Analysis of Bipolar Transistor Circuits"

IEEE Jour. Solid-State Circuits, SC-6, p. 14-19, February 1971.

4. R. D. Berry

"An Optimal Ordering of Electronic Circuit Equations for a Sparse Matrix Solution"

IEEE Trans. Circuit Theory, CT-18, p. 40-50, January 1971.

5. R. Rohrer, L. Nagel, R. Meyer, L. Weber

"Computationally Efficient Electronic-Circuit Noise Calculations"

IEEE Jour. Solid-State Circuits, SC-6, p. 204-213, August 1971.

6. S. W. Director

"Survery of Circuit Oriented Optimization Techniques"

IEEE Trans. Circuit Theory, CT-18, p. 3-10, January 1971.

7. B. A. Wooley

"Automated Design of DC-Coupled Monolithic Broadband Amplifiers"

IEEE Jour. Solid-State Circuits, SC-6, p. 24-34, February 1971.

8. C A Desoer, E S. Kuh

Basic Circuit Theory

New York: McGraw-Hill, 1969, Chapter 9, section 4, p. 392-396.

9. H. K. Gummel, H. C. Poon

"An Integrated Charge Control Model of Bipolar Transistors"

Bell System Tech. Jour. vol. 49, p. 827-852, May/June, 1970.

10. H. Shichman, D. A. Hodges

"Modeling and Simulation of Insulated-Gate Field-Effect Transistor Switching Circuits"

IEEE Jour. Solid-State Circuits, SC-3, p. 285-289, September 1968.

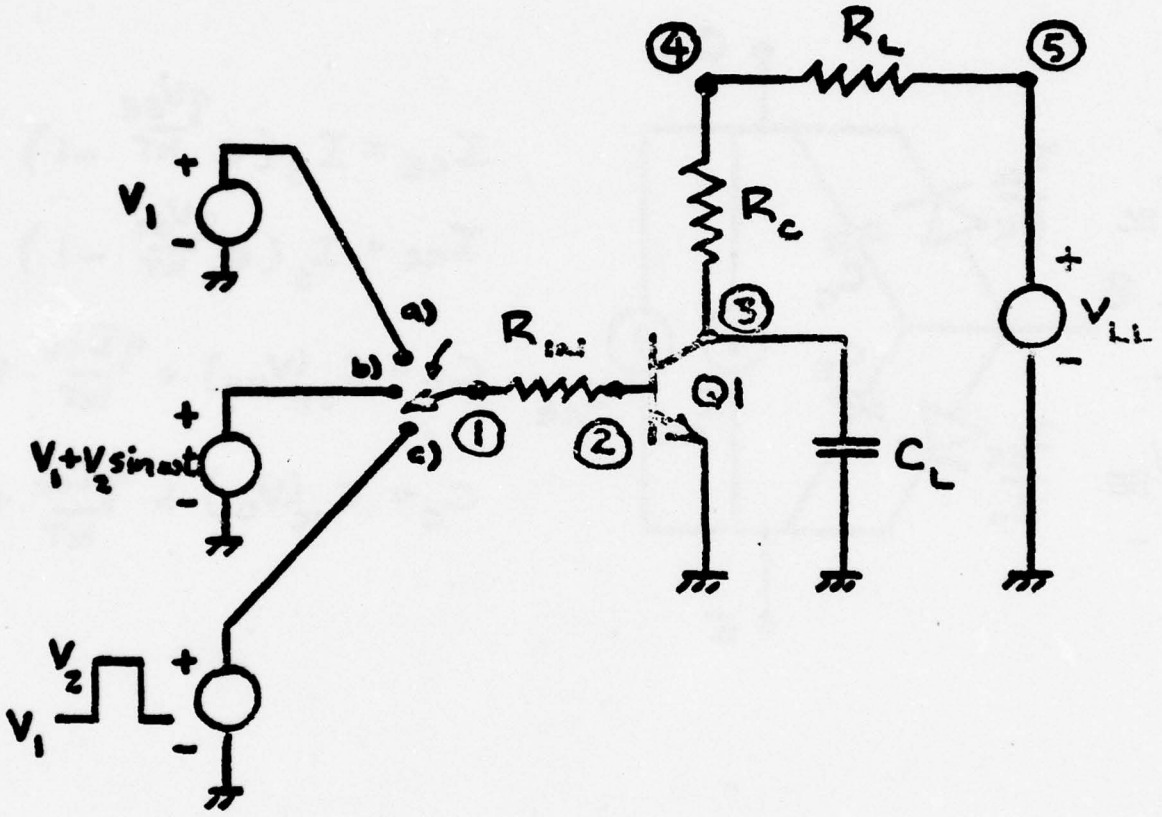
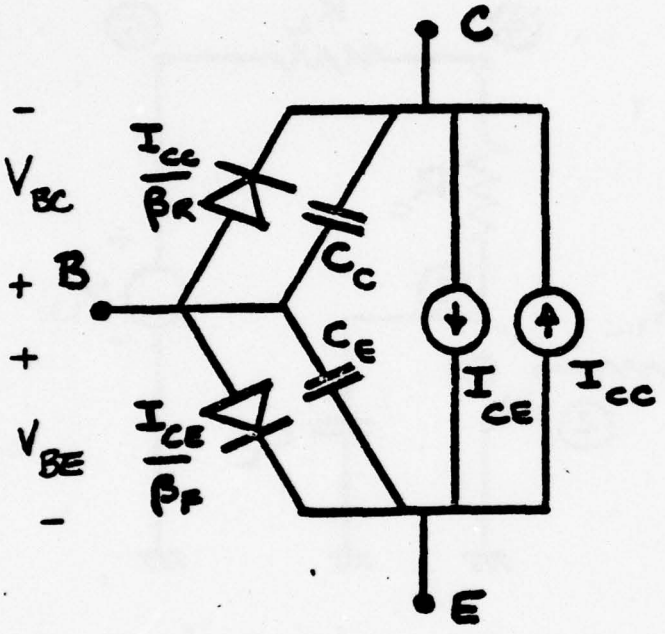


Figure 1. Bipolar Transistor Circuit Example to Illustrate Nodal Circuit Analysis using SPICE.



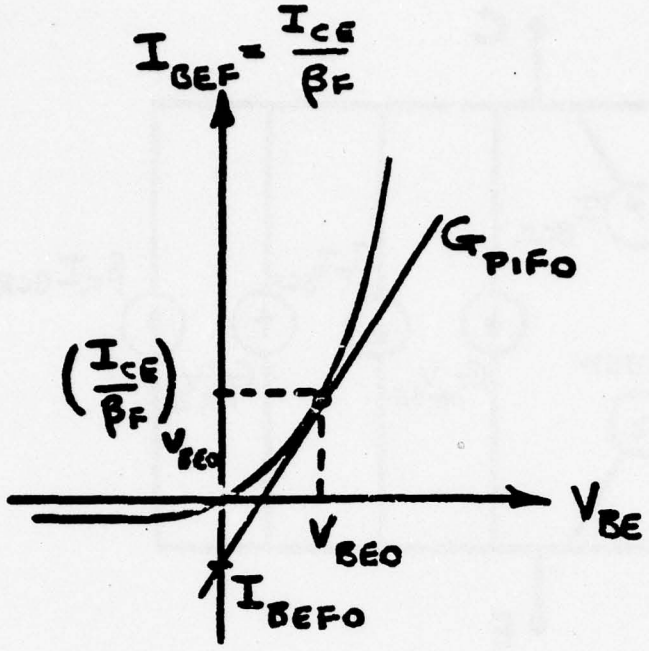
$$I_{CE} = I_S\left(e^{\frac{qV_{BE}}{kT}} - 1\right)$$

$$I_{CC} = I_S\left(e^{\frac{qV_{BC}}{kT}} - 1\right)$$

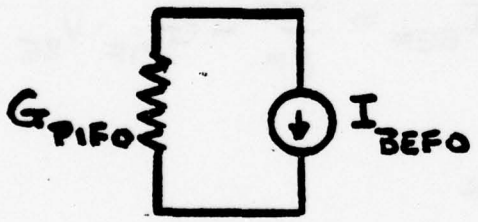
$$C_E = C_{JE}(V_{BE}) + \frac{qI_{CE}}{kT}\cdot\tau_F$$

$$C_C = C_{JC}(V_{BC}) + \frac{qI_{CC}}{kT}\cdot\tau_R$$

Figure 2 Nonlinear Bipolar Junction Transistor Model with Charge Storage Elements and Characteristic Equations.



(a)

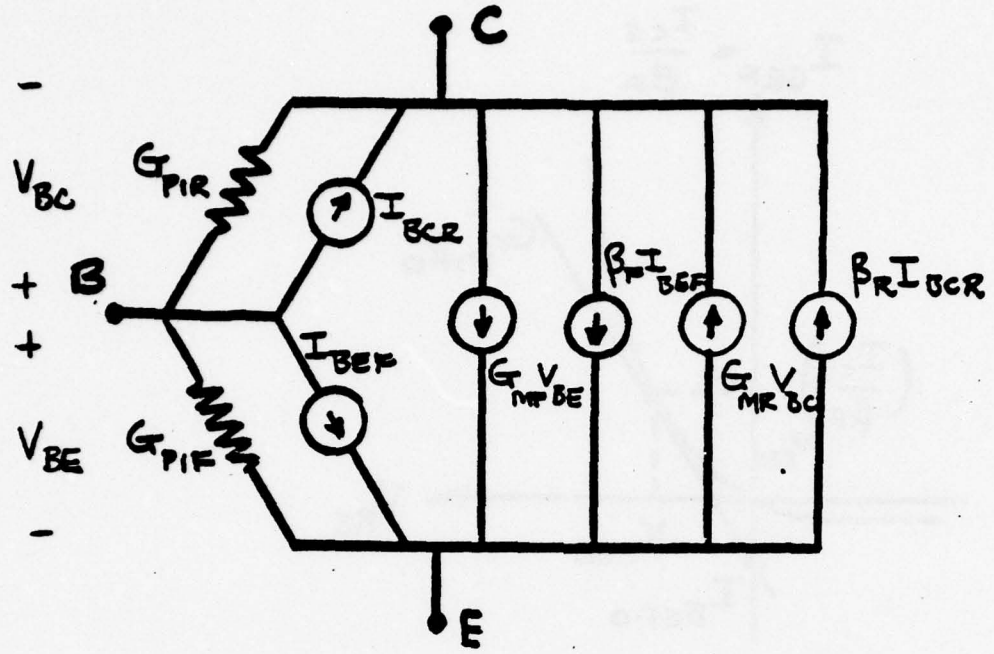


$$G_{PIFO} = \frac{\partial}{\partial V_{BE}} \left(\frac{I_{CE}}{\beta_F}\right)\Big|_{V_{BEO}}$$

$$I_{BEFO} = \left(\frac{I_{CE}}{\beta_F}\right)\Big|_{V_{BEO}} - G_{PIFO} \cdot V_{BEO}$$

(b)

Figure 3. a) The Exponential Diode Relationship for the Base Current due to I_{CE}/β_E with Notation used for Linearization,
b) The Linearized Model about V_{BEO}.



$$G_{PIR} = \frac{\partial}{\partial V_{BC}}\left(\frac{I_{CC}}{\beta_R}\right) \qquad I_{BCR} = \frac{I_{CC}}{\beta_R} - G_{PIR} \cdot V_{BC}$$

$$G_{PIF} = \frac{\partial}{\partial V_{BE}}\left(\frac{I_{CE}}{\beta_F}\right) \qquad I_{BEF} = \frac{I_{CE}}{\beta_F} - G_{PIF} \cdot V_{BE}$$

$$G_{MR} = \frac{\partial}{\partial V_{BC}}\left(I_{CC}\right) = \beta_R G_{PIR}$$

$$G_{MF} = \frac{\partial}{\partial V_{BE}}\left(I_{CE}\right) = \beta_F G_{PIF}$$

Figure 4. The Complete Linearized Bipolar Junction Transistor Model with Parameter Values to be Determined as Indicated.

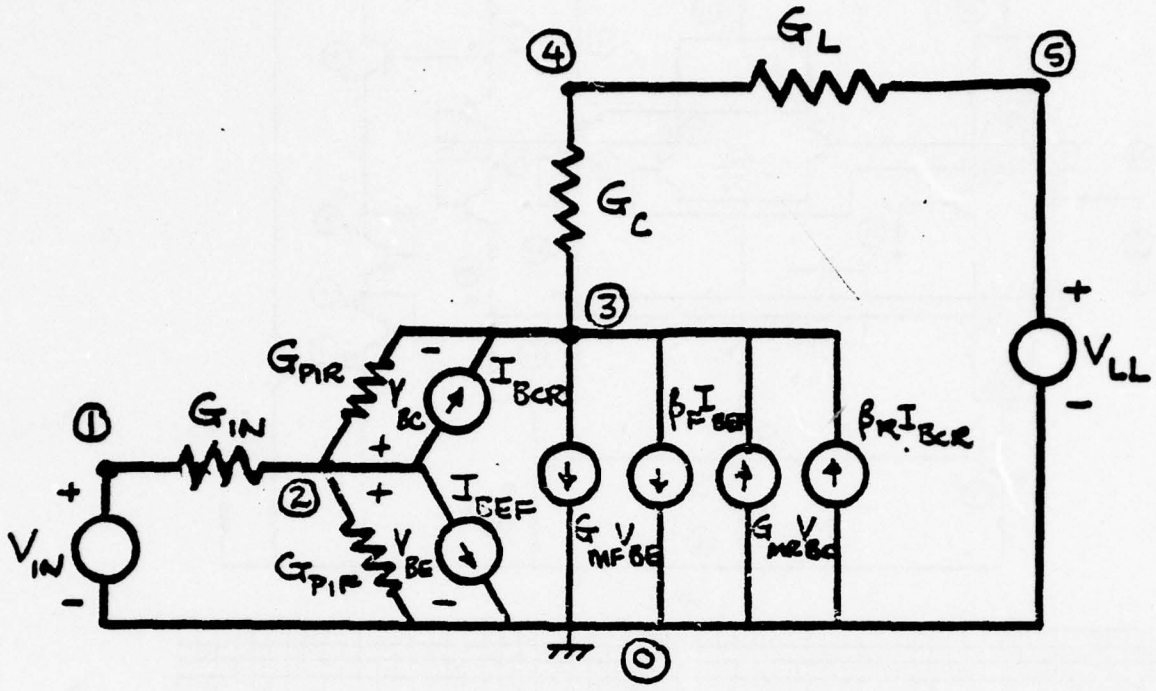


Figure 5. The Transistor Circuit from Figure 1 for dc conditions with Linearized Transistor Model.

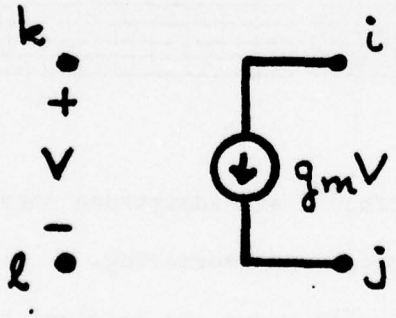
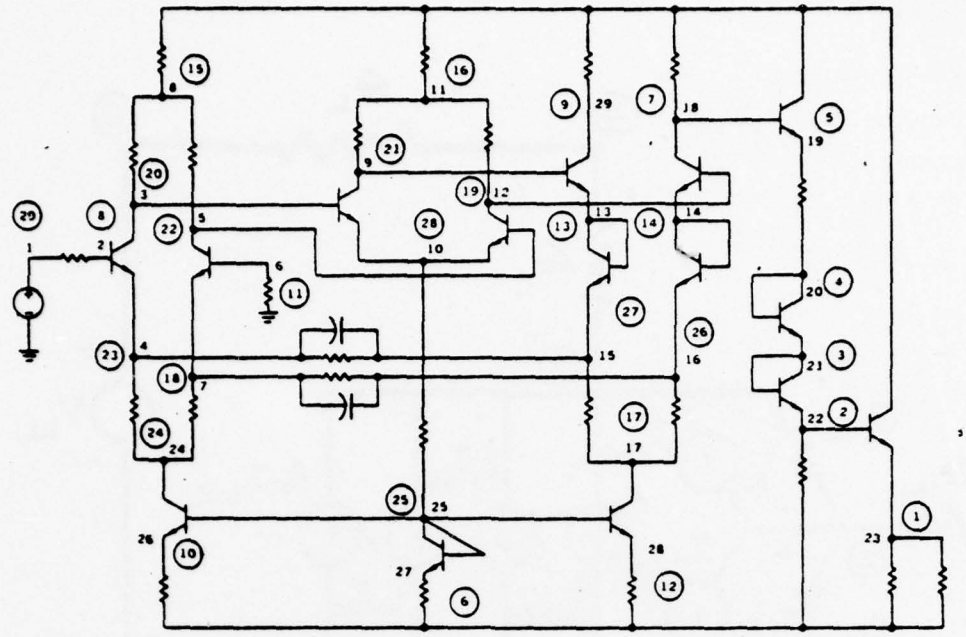
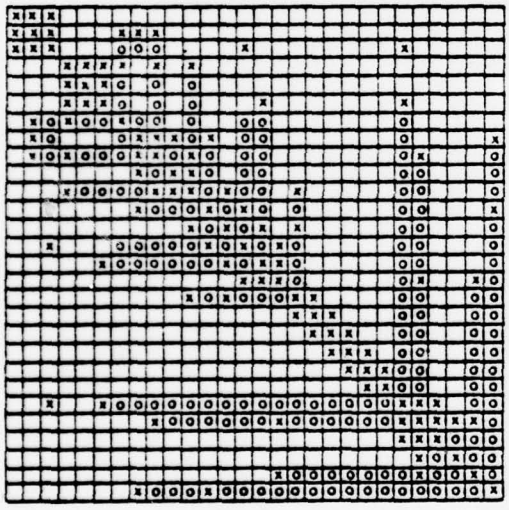


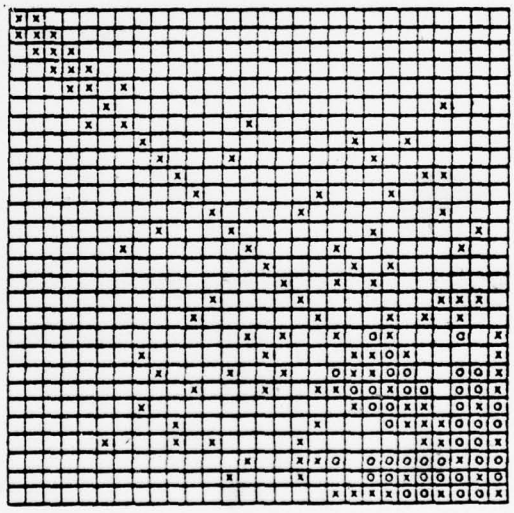
Figure 6. Voltage-controlled Current Source Model with Reference Directions and Node Numbers.



(a)



(b)



(c)

Figure 7. A circuit configuration[4] and admittance matrix structure to illustrate impact of sparse storage and reordering.

a) the circuit, uncircled nodes are original and circled indicate renumbered,

b) the matrix structure before reordering. The X's indicate entries, the O's fill-in.

c) the matrix structure after node reordering. Note the reduced fill-in.

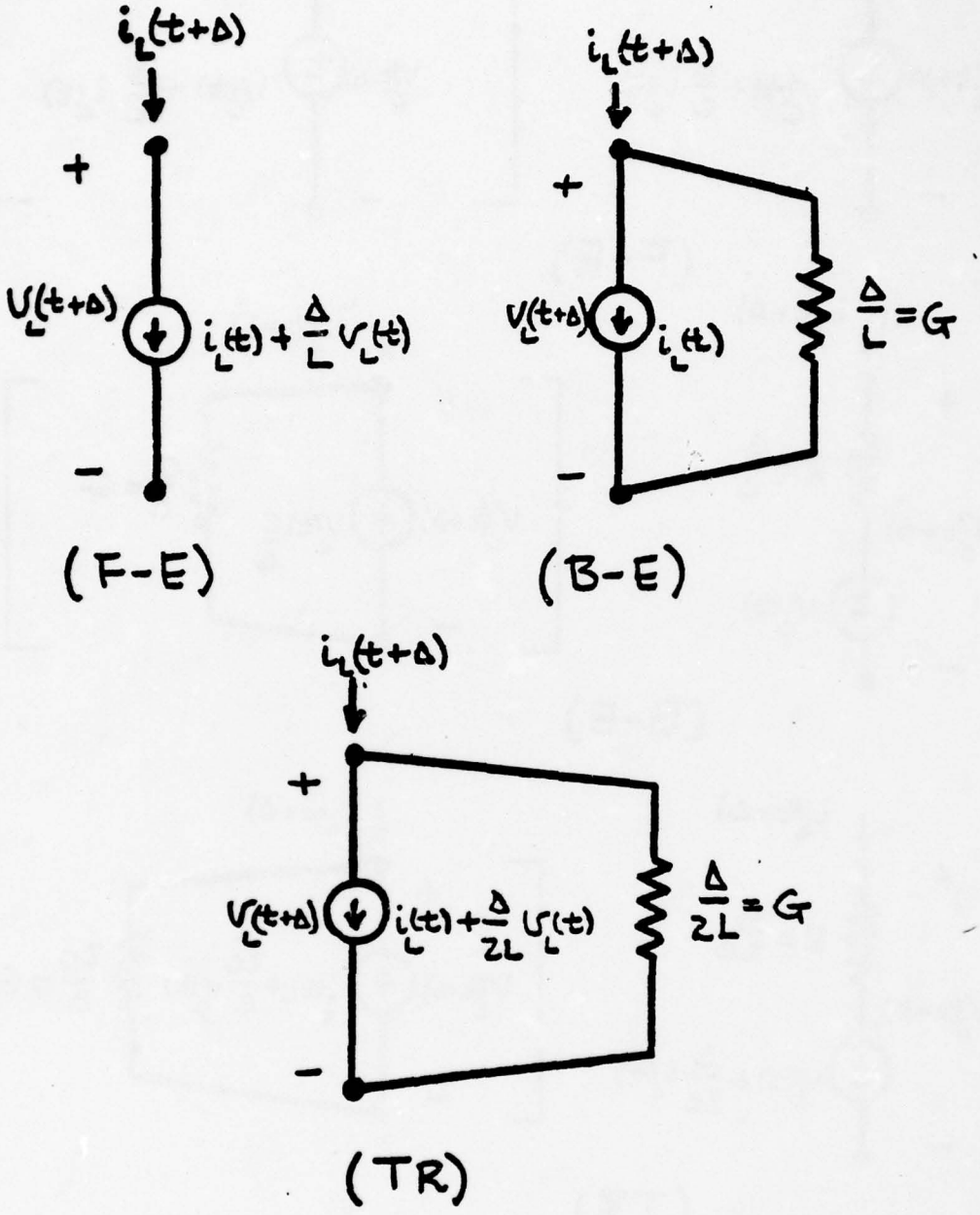


Figure 3 a) Equivalent Circuit Interpretations of the Integration Formula for an Inductor for Forward Euler (F-E), Backward Euler (B-E) and Trapezoidal (TR) Integration Rules.

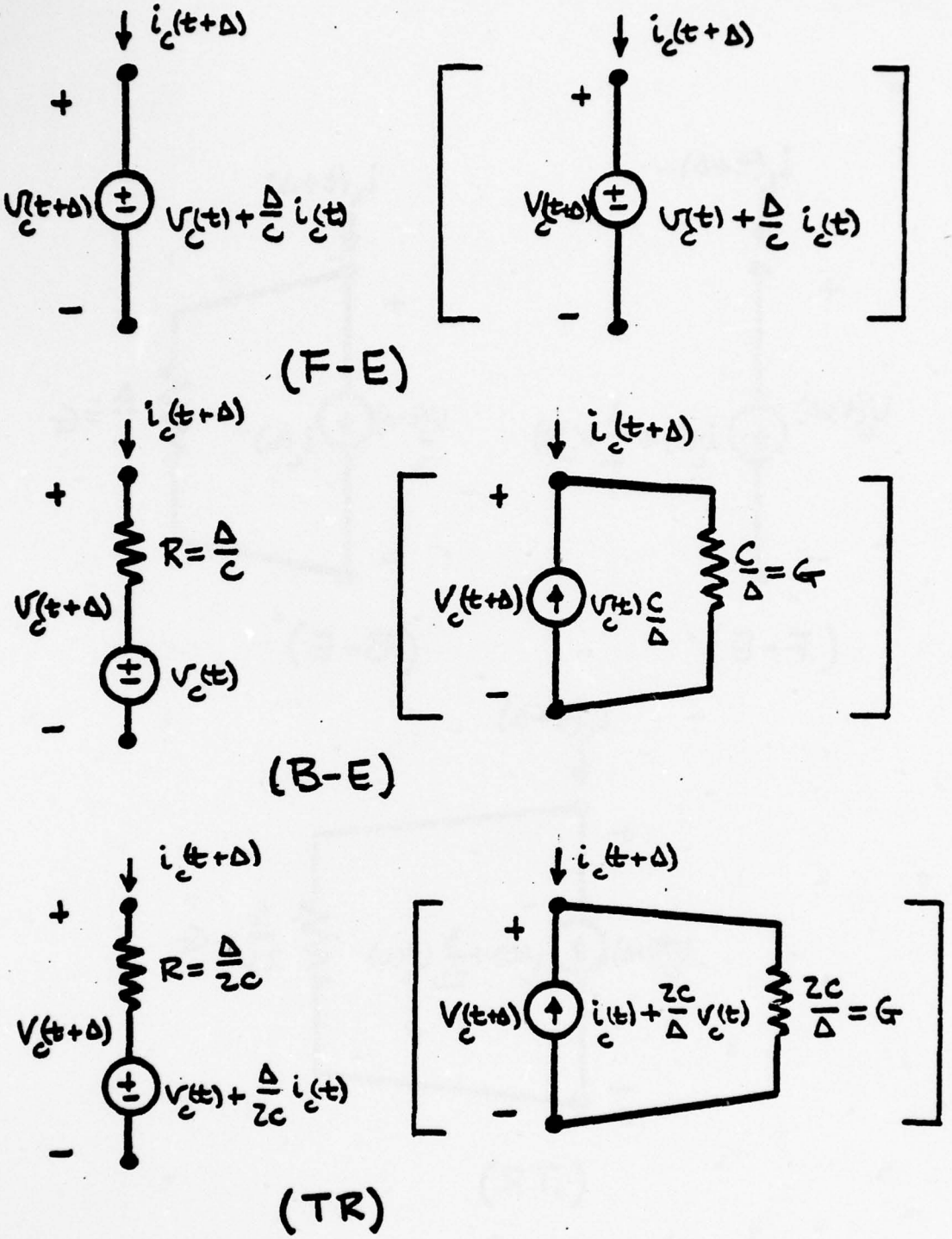


Figure 8 b) Equivalent Circuit Interpretation of the Integration Formula, for a Capacitor for Forward Euler (F-E), Backward Euler (B-E) and Trapezoidal (TR). The square bracketed circuits are the equivalent forms appropriate for nodal analysis.

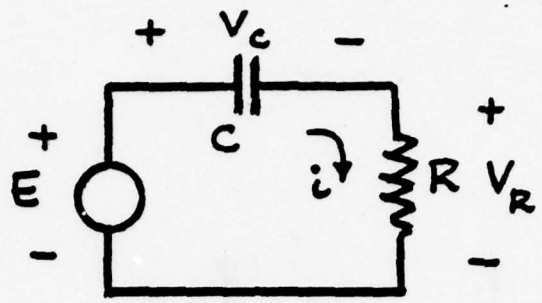


Figure 9 Circuit Example to Illustrate Accuracy and Stability of Integration Formulae

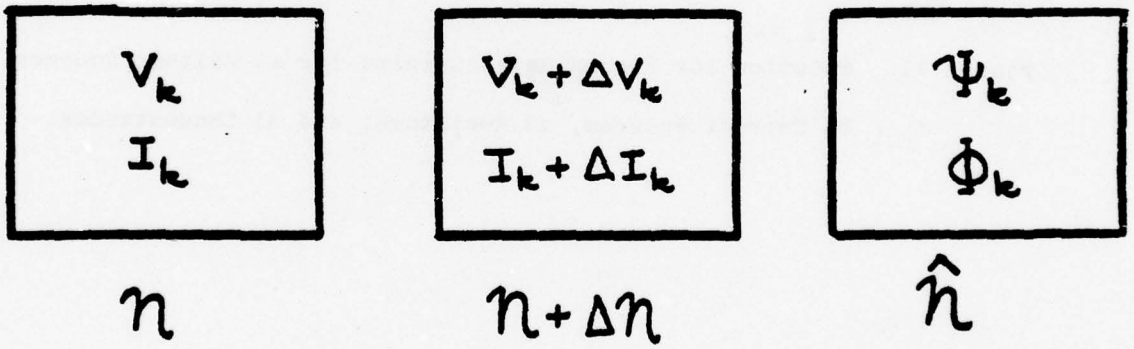


Figure 10. Three Networks $\eta, \hat{\eta}$ and $\eta + \Delta\eta$ which Satisfy Tellegen's Theorem. The Voltage-Current Definitions are Shown for Each Network.

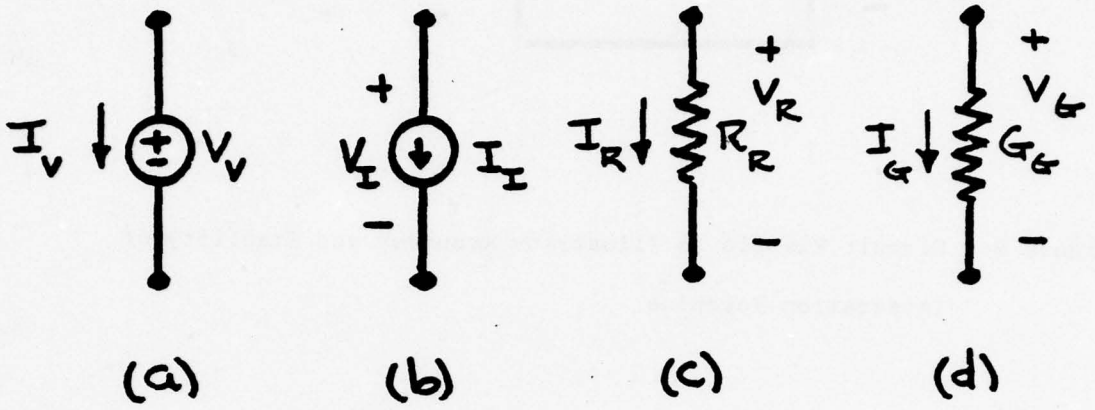


Figure 11. Notation for Branch Relationships for a) Voltage Sources, b) Current Sources, c) Resistors, and d) Conductances.

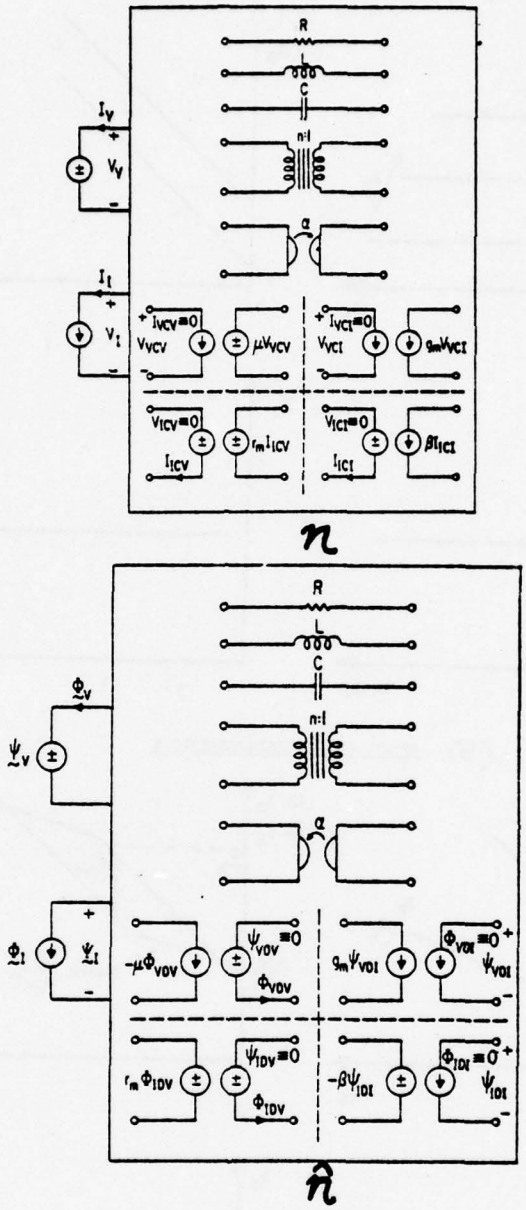


Figure 12. Symbolic representation of the η and $\hat{\eta}$ networks and the meaning of individual elements in each network (and correspondences).

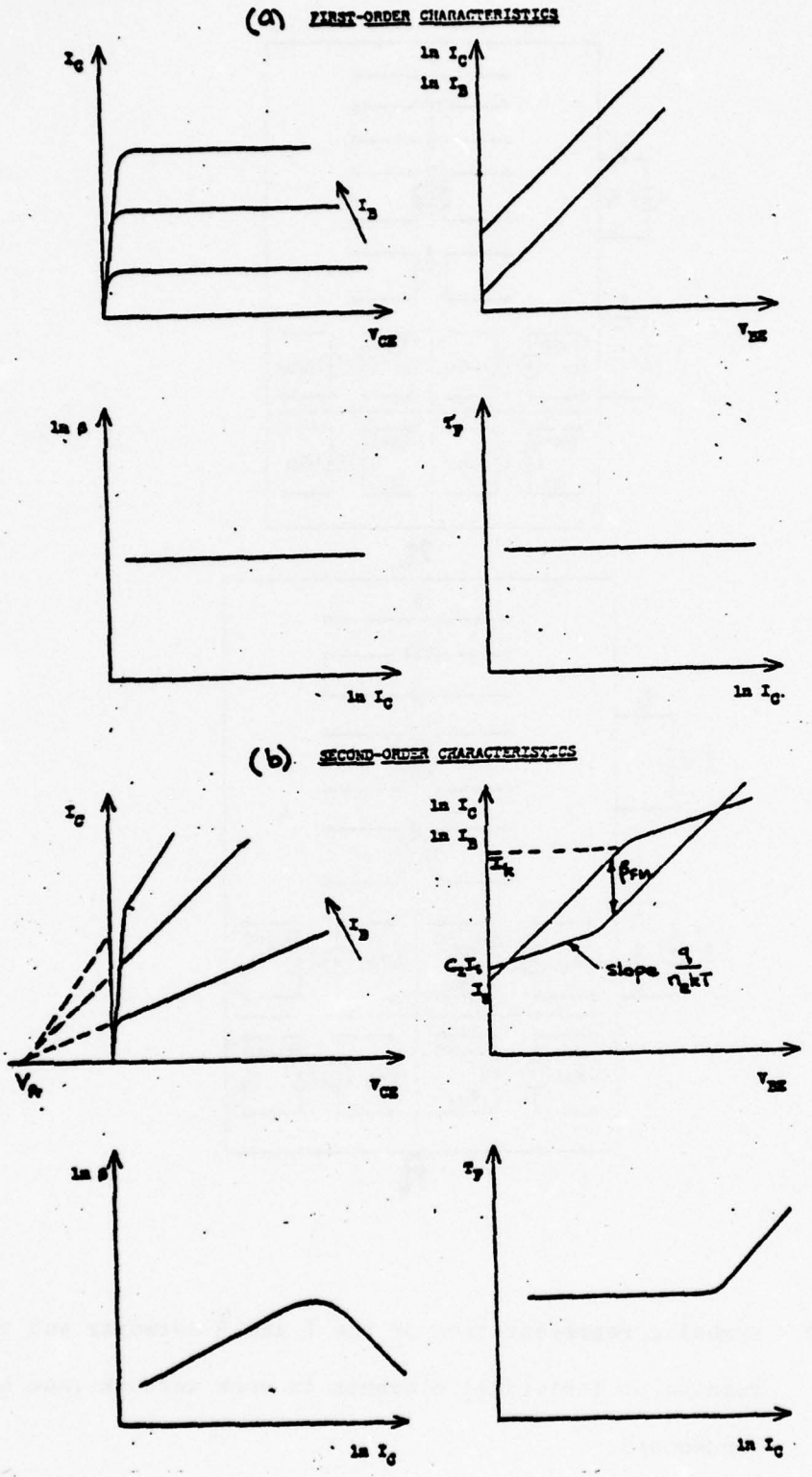


Figure 13. Bipolar Transistor Device Characteristics and Parameter Definiti

a) for the first-order (Ebers-Moll) model,

b) for the Gummel-Poon (second-order) model with parameters as defined.

APPENDIX I

Reprinted from the IEEE Transactions on Circuit Theory, Vol. CT-18, No. 1, January 1971.

Elements of Computer-Aided Circuit Analysis

WILLIAM J. McCALLA, MEMBER, IEEE, AND DONALD O. PEDERSON, FELLOW, IEEE

Abstract—A survey is made of the principal techniques, procedures, and routines that are used in present programs for computer-aided circuit analysis. Programs (simulators) are reviewed and selected features compared for the four major classes of circuit analysis: linear dc and ac, nonlinear dc, nonlinear transient, and linear pole zero.

I. Introduction

SEVERAL recent books and papers have cataloged and compared many existing and proposed computer-aided circuit analysis programs [1]–[11]. This paper is a survey of the principal techniques used in existing programs for the analysis of the four important circuit areas: linear dc and ac, nonlinear dc, nonlinear transient, and linear pole zero. The considerations brought out are based on over four years experience in using, modifying, and writing more than twenty programs.

This survey is organized as follows: first an overview is made comparing nodal and mixed or state-variable methods of formulating the circuit equations. Linear dc and ac analysis programs are then considered with emphasis placed on techniques for solving systems of linear equations. Attention is next given to nonlinear dc analysis programs and the iterative solution of nonlinear algebraic equations. The extension to nonlinear transient analysis follows with an introduction to several numerical integration routines used in the solution of the nonlinear differential equations. Finally, four linear pole-zero circuit analysis techniques and programs are considered. Throughout the paper a number of specific references on computer-aided circuit analysis and design are cited. In addition, several general computer-aided circuit analysis references are included [12]–[17].

II. Circuit Equation Formulation

Virtually all presently available circuit analysis programs start from the same point, an elemental circuit description supplied to the program via keyboard, punched cards, or an interactive graphic display console. This description of circuit elements and their interconnections is converted by the programs into a set of circuit equations. Of interest here are the two major approaches that are now used in formulating these equations. The first approach is the familiar nodal analysis while the second is the mixed or state-variable approach which derives from Bashkow's A-matrix formulation [18].

The implementations of these two approaches within the present generation of circuit analysis programs seem to have had a common origin in the transistor analysis program (TAP) [19]–[20] developed by IBM. TAP, though reasonably effective, was short lived and had such severe limitations that it was never made available outside of IBM. From TAP there evolved three programs: ECAP [21], which used nodal analysis, and NET-1 [22] and PREDICT [23], which both used a state-variable approach.

ECAP performs linear dc, linear ac, and piecewise-linear transient analyses and is still very much in use. It is considered in more detail shortly. NET-1, a nonlinear transient analysis program, is available in the IBM 7090/94 assembly languages MAP and FAP. Therefore, this program cannot be used simply in computer systems having monitor systems. NET-1 did, however, influence the development of the program CIRCUS [24]. This program is considered in more detail in the following paragraphs. PREDICT is also a nonlinear transient analysis program but includes no built-in device models and is incapable of automatically performing multiple analyses for several different sets of element values. In addition, it requires separate analysis runs for steady-state (dc) and transient solutions and suffers from a weak numerical integration routine. To overcome these and other deficiencies, the SCEPTRE [25] program, which is considered in the following paragraphs, was developed.

Nodal Analysis Formulation

For both linear and nonlinear (or piecewise-linear) circuit problems, nodal equations may be generated by little more than inspection. The relative simplicity of this approach contrasts sharply with the manipulations required by the state-variable approach. As an example, consider the case of a linear circuit. The nodal equations are of the form

$$Yv = i \tag{1}$$

where Y is the nodal admittance matrix, v is the vector of node voltages to be found, and i is a vector representing independent source currents. The term y_{ii} in Y represents the sum of the admittances of all the branches connected to node i; y_{ij} is the negative of the sum of the admittances of all branches connecting node i and node j; and i_k is the sum of all source currents entering node k. Thus if a resistor of value R connects nodes 5 and 7, $1/R$ is added to y_{55} and y_{77} and subtracted from y_{57} and y_{75}, while if a current source of

Manuscript received August 11, 1970. This work was supported in part by the Joint Services Electronics Program, under Grant AFOSR-68-1488, and in part by the U. S. Army Research Office, Durham, N. C., under Contract DAHCO4-67-C-0031.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, Calif. 94720.

strength I is directed from node 2 to node 3, I is subtracted from i_2 and added to i_3.

Voltage sources are usually handled in one of two ways. The first is to require that every voltage source appear in series with a resistor so that the source may be transformed to a Norton equivalent current source. The second approach is a generalization of the first but does not depend upon a series resistor. The approach is most easily introduced in terms of grounded voltage sources. The nodal equations are first assembled including all elements other than voltages sources. Columns of the admittance matrix corresponding to grounded voltage source nodes are then multiplied by the value of the source and the result is subtracted from both sides of (1). If the current through the voltage source is required, it may be treated as an unknown in place of the known voltage. Otherwise the equation representing the sum of the currents at the source node may be considered redundant and dropped, thereby reducing the number of unknowns. Floating voltage sources and controlled sources may be handled by similar column and row operations.

For nonlinear analysis, equations can be formulated in a manner similar to that used in the linear case. The equations may be written symbolically as

$$Y(v, t) = i(t). \tag{2}$$

The interpretation of (2) is that each equation represents a summing of current contributions at a node. It should be noted, however, that as now formulated the nonlinearities are restricted to be voltage rather than current dependent. Fortunately, semiconductor devices such as junction diodes and bipolar and field-effect transistors are of the voltage-controlled category, and the above restriction is usually not severe.

As mentioned previously, ECAP is based on nodal analysis formulation. Additional nonlinear analysis programs using nodal analysis include TRAC [26], MTRAC [27], SYSCAP [28], and BIAS-3 [29], while linear programs based on a nodal approach include NATFREQS [30], [31], and from LISA [32], [33] the programs ACCA, POLY, and TRFN.

State-Variable Analysis Formulation

The primary reason for using the state-variable approach is that it yields a set of first-order linear or nonlinear differential equations in a minimal set of unknowns. Deriving a set of such equations explicitly seemed a desirable objective for use with earlier numerical integration algorithms. As is brought out in the following paragraph, this is no longer essential.

From an elementary viewpoint, the state-variable formulation proceeds as follows [34]: a proper tree consisting of all voltage sources, as many capacitive branches and as few inductive branches as possible, and no current sources is selected. The state variables are chosen to be the capacitive tree-branch voltages or charges and inductive tree-link currents or fluxes. A fundamental cut-set equation is constructed for each capacitive tree branch and a fundamental loop equation for each inductive tree link. Independent sources are withdrawn as separate terms in the equations and the equations are normalized with respect to the element values associated with each state variable.

For a linear circuit, the preceding procedure results in a matrix equation of the form [35]

$$\dot{x} = Ax + Bu \tag{3}$$

where A is a coefficient matrix relating the state vector x to its derivative $\dot{x}$ and B is a coefficient matrix coupling the effects of the independent source vector u. Any other desired network variables can be expressed in terms of the state variables and independent sources. For the linear case, this procedure results in an equation of the form

$$y = Cx + Du \tag{4}$$

where y is a vector of desired output variables and C and D are again coefficient matrices.

For the nonlinear case, (3) and (4) are of the form

$$\dot{x} = A(x, u, t) \tag{5}$$

$$y = C(x, u, t). \tag{6}$$

An alternative description is based on extracting the nonlinear portion of the circuit from the linear portion. This allows the state equations (5) and (6) to be written in the following equivalent form [15]:

$$\dot{x} = Ax + Bu + B'w \tag{7}$$

$$y = Cx + Du + D'w \tag{8}$$

$$g(w) = Ex + Fu + F'w. \tag{9}$$

Again, A, B, B', C, D, D', E, F, and F' are coefficient matrices, while x and u remain the state variables and independent source vector, respectively. The vector w represents the controlling or independent variable set associated with the nonlinearities $g(w)$. In the state-variable approach, no assumption is made regarding voltage or current control.

While the equation formulation procedure outlined above applies in general, details vary from program to program. Both CIRCUS and SCEPTRE employ variations of Bryant's method [36], [37] as modified by Wilson and Massena [38]. Other programs relying on the state-variable approach include CORNAP [39], [40], BELAC [41], and CIRPAC [42], [43]. In all cases, the formulation procedures entail extensive matrix manipulations. The bookkeeping associated with these manipulations usually requires considerable amounts of core storage. Loops of capacitors and voltage sources and cut-sets of inductors and current sources are accommodated by additional manipulations of the state equations.

III. Linear DC and AC Analysis

The first of the four circuit analysis areas to be treated is linear dc and ac analysis. The desired outputs are the various voltages and currents within a circuit, possibly as a function of frequency. As virtually all programs of this category use a nodal analysis formulation, it is assumed that the circuit equations to be solved have the form of (1). The major tech-

niques presently used to solve such systems of linear equations include Gaussian elimination, pivoting, LU factorization, and sparse matrix methods. These techniques are outlined in the following paragraphs and are followed by a comparison of the features of several available analysis programs.

A dc analysis can be considered the special case of an ac analysis performed at zero frequency. However, such an approach is usually not used for two reasons: first, if the formulation proceeds on a nodal admittance basis, the infinite admittance represented by an inductor at zero frequency requires special consideration. Second, where only resistive elements and independent and controlled sources are present, the circuit equations contain only real coefficients and hence may be solved on a computer using only real variables. If complex variables are used, the computation time may be increased by as much as a factor of 4 because multiplication and division (both considered long operations as compared with the short operations of addition and subtraction) require more time than for long operations with real variables.

Gaussian Elimination

With the distinction in mind that the system of nodal equations (1) involves only real variables in the dc case and complex variables in the ac case, the solution of (1) can be written

$$v = Y^{-1}i \tag{10}$$

where Y^{-1} is the inverse of the nodal admittance matrix. Computationally, the efficiency of this approach can be evaluated in terms of the number of long operations required. It can be shown that the number of long operations required to invert an $n \times n$ matrix is n^3, while the number of long operations required to compute its product with a vector of dimension n is n^2. Thus the total number of long operations required by this approach is

$$n^3 + n^2.$$

The number of long operations required to obtain a solution to (1) can be reduced by a factor of 3 using Gaussian elimination [44].

This procedure consists of two steps. The first step consists of converting the nodal admittance matrix Y to an equivalent upper triangular matrix, i.e., a matrix with all zero elements below the diagonal. The second step which is referred to as back substitution, consists of solving the nth equation containing only v_n for v_n, the $n-1$st equation for v_{n-1} in terms of v_n, etc. Gaussian elimination is illustrated in (11) for a third-order system

$$\begin{matrix}\mathcal{E}_1^{(1)}:\\ \mathcal{E}_2^{(1)}:\\ \mathcal{E}_3^{(1)}:\end{matrix}\begin{bmatrix} y_{11}^{(1)} & y_{12}^{(1)} & y_{13}^{(1)} \\ y_{21}^{(1)} & y_{22}^{(1)} & y_{23}^{(1)} \\ y_{31}^{(1)} & y_{32}^{(1)} & y_{33}^{(1)} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} i_1^{(1)} \\ i_2^{(1)} \\ i_3^{(1)} \end{bmatrix} \tag{11}$$

where the superscript 1 indicates the initial system of equations. As indicated previously, the unknown v_1 is eliminated from equations $\mathcal{E}_2^{(1)}$ and $\mathcal{E}_3^{(1)}$ by subtracting $(y_{21}^{(1)}/y_{11}^{(1)})\mathcal{E}_1^{(1)}$ from $\mathcal{E}_2^{(1)}$ and $(y_{31}^{(1)}/y_{11}^{(1)})\mathcal{E}_1^{(1)}$ from $\mathcal{E}_3^{(1)}$. Symbolically this transformation can be represented by the equations

$$\begin{aligned}\mathcal{E}_1^{(2)} &= \mathcal{E}_1^{(1)} \\ \mathcal{E}_2^{(2)} &= \mathcal{E}_2^{(1)} - \frac{y_{21}^{(1)}}{y_{11}^{(1)}}\mathcal{E}_1^{(1)} \\ \mathcal{E}_3^{(2)} &= \mathcal{E}_3^{(1)} - \frac{y_{31}^{(1)}}{y_{11}^{(1)}}\mathcal{E}_1^{(1)}\end{aligned} \tag{12}$$

which yields the system

$$\begin{matrix}\mathcal{E}_1^{(1)}:\\ \mathcal{E}_2^{(2)}:\\ \mathcal{E}_3^{(2)}:\end{matrix}\begin{bmatrix} y_{11}^{(1)} & y_{12}^{(1)} & y_{13}^{(1)} \\ 0 & y_{22}^{(2)} & y_{23}^{(2)} \\ 0 & y_{32}^{(2)} & y_{33}^{(2)} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} i_1^{(1)} \\ i_2^{(2)} \\ i_3^{(2)} \end{bmatrix}. \tag{13}$$

Finally, the unknown v_2 is eliminated from $\mathcal{E}_3^{(2)}$ by subtracting $(y_{32}^{(2)}/y_{22}^{(2)})\mathcal{E}_2^{(2)}$ from $\mathcal{E}_3^{(2)}$, thus obtaining

$$\begin{aligned}\mathcal{E}_1^{(3)} &= \mathcal{E}_1^{(2)} = \mathcal{E}_1^{(1)} \\ \mathcal{E}_2^{(3)} &= \mathcal{E}_2^{(2)} \\ \mathcal{E}_3^{(3)} &= \mathcal{E}_3^{(2)} - \frac{y_{32}^{(2)}}{y_{22}^{(2)}}\mathcal{E}_2^{(2)}\end{aligned} \tag{14}$$

resulting in the triangularized system

$$\begin{matrix}\mathcal{E}_1^{(1)}:\\ \mathcal{E}_2^{(2)}:\\ \mathcal{E}_3^{(3)}:\end{matrix}\begin{bmatrix} y_{11}^{(1)} & y_{12}^{(1)} & y_{13}^{(1)} \\ 0 & y_{22}^{(2)} & y_{23}^{(2)} \\ 0 & 0 & y_{33}^{(3)} \end{bmatrix}\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} i_1^{(1)} \\ i_2^{(2)} \\ i_3^{(3)} \end{bmatrix}. \tag{15}$$

This completes the first step in the Gaussian elimination procedure. Back substitution is now performed to obtain the final solution as follows:

$$\begin{aligned} v_3 &= \frac{i_3^{(3)}}{y_{33}^{(3)}} \\ v_2 &= \frac{(i_2^{(2)} - y_{23}^{(2)}v_3)}{y_{22}^{(2)}} \\ v_1 &= \frac{(i_1^{(1)} - y_{13}^{(1)}v_3 - y_{12}^{(1)}v_2)}{y_{11}^{(1)}}.\end{aligned} \tag{16}$$

Careful enumeration shows that for an nth order system the number of long operations required by Gaussian elimination is

$$\frac{n^3}{3} + n^2 - \frac{n}{3}.$$

LU Transformation

A modification of Gaussian elimination which is useful when more than one source or right-hand-side vector i is to be considered is the LU transformation [44]. This procedure consists of partitioning Y into an upper triangular matrix U and a lower triangular matrix L (usually with ones on the diagonal) such that

$$LU = Y. \tag{17}$$

This technique is illustrated shortly. The resulting system is solved in two stages. First,

$$Uv = L^{-1}i = i^* \tag{18}$$

and secondly,

$$v = U^{-1} i^*. \tag{19}$$

In this case since both L and U are triangular, L^{-1} and U^{-1} are trivial to compute. Note that (19) is the same back substitution performed during Gaussian elimination, while (18) is also equivalent to back substitution. To obtain U the same reduction procedure is performed as in Gaussian elimination. Thus, in terms of the earlier third-order example,

$$U = \begin{bmatrix} y_{11}^{(1)} & y_{12}^{(1)} & y_{13}^{(1)} \\ 0 & y_{22}^{(2)} & y_{23}^{(2)} \\ 0 & 0 & y_{33}^{(3)} \end{bmatrix}. \tag{20}$$

Similarly, L is found to be a byproduct of the triangularization procedure. For the example,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \dfrac{y_{21}^{(1)}}{y_{11}^{(1)}} & 1 & 0 \\ \dfrac{y_{31}^{(1)}}{y_{11}^{(1)}} & \dfrac{y_{32}^{(2)}}{y_{22}^{(2)}} & 1 \end{bmatrix}. \tag{21}$$

The elements of L are the coefficients in (12) and (14).

A convenient aspect of the preceding procedure is that since the diagonal elements of L are known to be 1's, L and U may share the same memory locations originally assigned to Y. Note further that the long operations count is the same as for Gaussian elimination. Once L and U have been computed, (18) and (19) can be applied repeatedly for different source vectors i. For m source vectors, the total long operations count becomes

$$\frac{n^3}{3} + mn^2 - \frac{n}{3}.$$

Pivoting

Further methods of reducing core requirements and operations counts are brought out below. First, however, mention should be made of error control. As can be seen from (12) and (14), a division and a subtraction are required at each step in the triangularization process. Suppose a computer which represents a number accurately to d digits, performs arithmetic operations correctly to $2d$ digits, and then rounds the result to d digits. It can be shown [44] that the absolute error resulting from such arithmetic operations is given by $1+\phi\ 10^{-d}$ where $0 \le \phi \le 5$. In the floating-point notation, the operations of subtraction and division yield results where magnitudes are less than those of the operands. The relative error in the result is larger; thus division and subtraction steps reduce accuracy. The lower accuracy is felt more severely on computers with smaller word lengths where d is reduced. One obvious solution is to declare all variables to be double precision and perform all operations in double precision. However, this significantly increases both time and core memory requirements.

One compromise which can be made is described by Ralston [45]. He shows that the critical steps involving divisions and accumulation of partial sums can be performed in double precision at the expense of only one additional double precision vector of dimension n. The only major added expense is computation time.

The most commonly used approach of reducing error (used by Ralston with the preceding method) is pivoting [45]. Partial pivoting amounts to scanning the elements of the ith column of the matrix below the diagonal at the ith step in the reduction and determining the element of largest magnitude. The row of this element is exchanged with the ith row and the reduction is continued. For the example considered previously, if at the second step $|y_{32}^{(2)}| > |y_{22}^{(2)}|$, $y_{32}^{(2)}$ would be chosen as the pivot element, $\mathcal{E}_3^{(2)}$ and $\mathcal{E}_2^{(2)}$ would be exchanged and $y_{22}^{(2)}$ would be eliminated from $\mathcal{E}_2^{(2)}$ to obtain $\mathcal{E}_2^{(3)}$. This technique tends to reduce the magnitude of the term being subtracted at each step and thus improve accuracy. Further, it preserves column order and hence the order in the solution vector v.

Complete pivoting involves finding the largest element in the as yet unreduced $(n-i)\times(n-i)$ submatrix at the ith step, exchanging rows and columns such that it appears on the diagonal in the ith row and column. Since column order is not preserved except at the expense of additional bookkeeping, complete pivoting is seldom used.

Sparse Matrix Techniques

In view of the n^3 dependence of the long operations count, computation time can be expected to increase significantly as larger circuits with more nodes are considered. Several recent papers [46]–[48] have focused attention on taking advantage of sparsity in the nodal admittance matrix. There are basically three associated economies. First, efficient means have been found by which only the nonzero entries of the matrix need be stored, thus effecting a savings in core memory. Second, it is possible to process only the nonzero entries at each step in the triangular reduction. Finally, the order in which variables are eliminated can be chosen to preserve sparsity. The long operations count and computation time are then reduced. This savings becomes even more significant when the same equations must be solved many times, as in multifrequency analysis. The optimal order for eliminating variables need only be determined once.

By way of illustration [86], in a new program developed at the University of California, Berkeley, by Prof. R. Rohrer and his students, the third technique of optimal ordering together with nonzero storage leads to a long operation count more closely proportional to n than to n^3. In the analysis of a typical operational amplifier of 22 nodes, the number of long operations was reduced from 2660, using Gaussian elimination, to 130.

Linear DC and AC Analysis Programs

Many linear analysis programs are available including ECAP, the ACCA portion of LISA, and ROHRERX.[1] As previously mentioned, these programs are based on a nodal analysis formulation. Both ACCA and ROHRERX have free

[1] ROHRERX is a program written and developed in 1969 by the IC Group, Electronics Research Laboratory, University of California, Berkeley.

format input languages as do certain time-sharing versions of ECAP. The primary advantages of ECAP are that it is well documented [50] and is available through many commercial time-sharing companies. Its disadvantages are that it is cumbersome to modify and the original input language is cumbersome to use.

ACCA, as part of the larger LISA package, shares a general input routine and has the advantage of being able to interact somewhat with other portions of LISA. Its disadvantage is that, if used as part of LISA, it is expensive to load. ROHRERX does make partial use of sparse matrix techniques and is therefore relatively fast. It presently suffers, however, from a lack of documentation.

At the present time efforts are being made to decrease computation time and to incorporate sensitivity and noise performance using the adjoint network [51]. This work is of major importance in the development of several computer-aided circuit design packages [52]–[54].

IV. Nonlinear DC Analysis

Most nonlinear dc analysis programs are incorporated into more general transient analysis programs. Equation formulation approaches are divided between nodal and state-variable analysis. In either approach, the nonlinear dc analysis problem reduces to one of solving simultaneous nonlinear algebraic equations. As described in the following paragraphs, iterative methods are used. Of particular importance is the convergence properties of the solution algorithm. Several approaches which are used in different programs to improve convergence are examined in this section. Finally, various programs are again compared.

Functional Iteration

The solution method used by virtually all of the presently available nonlinear dc analysis programs is based on the Newton–Raphson iteration technique. It is one of a broad class of techniques known collectively as functional iteration methods [44].

Given a set of nonlinear equations of the form of (2), for the dc case the equations may be expressed

$$g(v) = 0. \tag{22}$$

The solution technique is to start from some initial set of values $v^{(0)}$ and to generate a sequence of iterates $\cdots v^{(\nu-1)}$, $v^{(\nu)}$, $v^{(\nu+1)}, \cdots$ which converge to the solution $\hat{v}$.

Newton–Raphson iteration is most easily introduced by considering the case of a single nonlinear equation

$$g(v) = 0. \tag{23}$$

The function $g(v)$ can be expanded about some point v_0 in a Taylor series to obtain

$$g(v) = g(v_0) + (v - v_0)g'(v_0) + \cdots = 0 \tag{24}$$

where the prime denotes differentiation with respect to v.

If only first-order terms are retained, a rearrangement of (24) yields

$$v = v_0 - \frac{g(v_0)}{g'(v_0)}. \tag{25}$$

The form of (25) suggests that a sequence of iterates might be generated by the following:

$$v^{(\nu+1)} = v^{(\nu)} - \frac{g(v^{(\nu)})}{g'(v^{(\nu)})}. \tag{26}$$

Equation (26) is the Newton–Raphson iteration function for the scalar case. Note that at the solution $\hat{v}$, $g(\hat{v})=0$ and $v^{(\nu+1)}=v^{(\nu)}$ as would be expected. The geometrical interpretation of (26) is illustrated in Fig. 1 for the simple case of a current source driving an ideal diode. The line tangent to the nonlinearity at the point $(v^{(\nu)}, g(v^{(\nu)}))$ has the slope $g'(v^{(\nu)})$. Its intercept with the voltage axis defines the next voltage iterate in the sequence as shown in the figure.

The generalization of the Newton–Raphson procedure to a system of n equations is given by

$$v^{(\nu+1)} = v^{(\nu)} - J^{-1}(v^{(\nu)})g(v^{(\nu)}) \tag{27}$$

where the Jacobian $J(v)$ of the function $g(v)$ is given by

$$J(v) = \begin{bmatrix} \frac{\partial g_1}{\partial v_1} & \frac{\partial g_1}{\partial v_2} & \cdots & \frac{\partial g_1}{\partial v_n} \\ \vdots & & & \vdots \\ \frac{\partial g_n}{\partial v_1} & \cdots & & \frac{\partial g_n}{\partial v_n} \end{bmatrix}. \tag{28}$$

A physical interpretation of the elements of $J(v)$ is brought out below.

The direct application of (27) necessitates computing the inverse of the $n \times n$ Jacobian matrix. As indicated previously, the operation count for inverting a matrix and multiplying the result by a vector is $n^3 + n^2$.

An alternative procedure for obtaining new iterates is to solve the linear system of equations

$$J(v^{(\nu)})(v^{(\nu)} - v^{(\nu+1)}) = g(v^{(\nu)}). \tag{29}$$

A second alternative procedure often used with nodal analysis is to employ the system of equations

$$J(v^{(\nu)})v^{(\nu+1)} = J(v^{(\nu)})v^{(\nu)} - g(v^{(\nu)}). \tag{30}$$

The right-hand side of (30) is found to have a particularly simple interpretation as also brought out below.

Gaussian elimination applied to either (29) or (30) reduces the long operation count to $(n^3/3)+n^2-(n/3)$. An even greater advantage is obtainable using sparse matrix methods. The locations of the nonzero elements of the Jacobian matrix are fixed by the circuit topology and remain unchanged from iteration to iteration. The additional time required on the first iteration to record the nonzero structure and determine the optimal variable elimination order is small in comparison to the total computation time required as the number of iterations becomes large.

Newton–Raphson Applied To Nonlinear DC Analysis

A physical interpretation of the Jacobian matrix and the Newton–Raphson method can be made using the diode circuit of Fig. 2(a). The exponential nonlinearity of the diode is linearized about some trial solution voltage V_0.

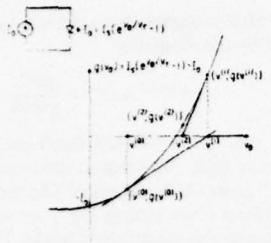


Fig. 1. Newton–Raphson iteration.

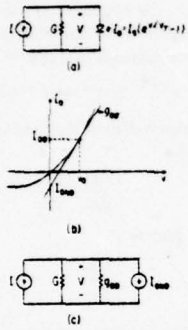


Fig. 2. Nonlinear dc analysis. (a) Diode circuit. (b) Linearized diode approximation. (c) Linearized circuit model.

This is equivalent to a Taylor series expansion as indicated previously where only first-order terms are retained. The expansion is of the form

$$I_D = I_D\Big|_{V=V_0} + (V - V_0)\frac{\partial I_D}{\partial v}\Big|_{V=V_0} \tag{31}$$

$$= I_s(e^{V_0/V_T} - 1) + (V - V_0)\frac{I_s}{V_T}e^{V_0/V_T} \tag{32}$$

$$= I_{DO} + (V - V_0)g_{DO} \tag{33}$$

where I_{DO} is recognized as the current through the diode corresponding to the voltage V_0, and g_{DO} is recognized as the dynamic conductance corresponding to the voltage V_0. Since the diode characteristic as described by (33) has now been linearized, the diode may be modeled in terms of a Norton equivalent current source I_{DNO} in parallel with the conductance g_{DO}. As can be seen from Fig. 2(b), I_{DNO} is given by

$$I_{DNO} = I_{DO} - g_{DO}V_0. \tag{34}$$

Hence, (32) may be written in terms of I_{DNO} as

$$I_D = g_{DO}V + I_{DNO}. \tag{35}$$

The nodal equation for the complete linearized circuit of Fig. 2(c) is

$$(G + g_{DO})V = I - I_{DNO}. \tag{36}$$

In terms of iterate values

$$(G + g_D^{(\nu)})V^{(\nu+1)} = I - I_{DN}^{(\nu)}. \tag{37}$$

If (30) is compared with (37) the physical interpretations of the Jacobian and the right-hand side of (30) become more apparent. The Jacobian consists of the nodal conductance matrix of the linear elements of the circuit together with the linearized conductances associated with each nonlinear circuit element. The vector on the right-hand side of (30) consists of independent source currents and the Norton equivalent source currents associated with each nonlinear circuit element. Thus at each iteration in a nodal analysis, the linearized conductances and Norton equivalent source currents must be recomputed and the linearized nodal conductance equations reassembled.

The application of Newton–Raphson iteration to the state-variable approach is also straightforward once all required coefficient matrices have been assembled. Recall that the equations formulated by the state-variable method can be put in the form of (7)–(9). The state vector x may represent capacitor voltages and inductor currents in which case the steady-state dc solution is characterized by $\dot{x} = o$. This corresponds to setting the current through capacitors and voltages across inductors to zero. Next, (7) may be solved for x in terms of w and the result substituted into (9) to obtain a system of nonlinear equations in terms of w. Once w has been obtained x and y can also be obtained. Note that since the coefficient matrices involved are constant, the initial manipulation necessary to obtain the single system of nonlinear equations in w need only be performed once.

Convergence

For the nonlinear dc analysis approach just outlined, the problem of convergence to a solution is now considered. Proofs that an algorithm will converge depend upon a priori knowledge of an initial guess sufficiently close to the solution. Because this knowledge is usually not present in a nonlinear dc analysis, all techniques incorporated into analysis programs for improving convergence of the Newton–Raphson iteration technique are supported by purely empirical justification.

The exponential nonlinearities usually associated with diodes and bipolar transistors are single-valued monotonically increasing continuous functions. However, these expressions are strongly increasing functions. For large reverse bias, the slope approaches zero, while for large forward bias the exponential tends to infinity. Convergence may be slowed or the iteration procedure may be stopped when numbers exceed a computer-imposed limit. An approach commonly used to prevent such overflows is illustrated in Fig. 3 again for a simple diode characteristic. As previously indicated, at a trial junction voltage $v^{(\nu)}$, the characteristic is modeled in terms of a linearized approximation as shown. A new iterate value $v^{(\nu+1)}$ greater than $v^{(\nu)}$ must correspond to a solution on the linearized characteristic at the point 1′. Two choices for a new trial operating point are immediately available. The first is to update

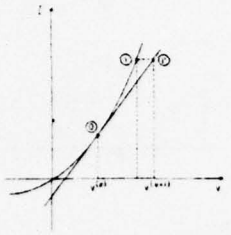


Fig. 3. Selection of new trial operating point.

with voltage by proceeding vertically to a new point on the exponential characteristic. This is the standard Newton–Raphson iteration and can be carried out by a straightforward evaluation of the exponential function. The second choice, which prevents the overflow problem, is to update with current by moving back horizontally to the exponential characteristic. This is done by computing the logarithm of the current corresponding to the point 1′ and results in the selection of point 1 as the new trial linearization point. Note that this procedure, which is particularly suited for junction diode and bipolar transistor circuits, can be used with both nodal analysis and state-variable analysis. This modified Newton–Raphson method can be used with other nonlinear devices but does require that they are described by functions which have an explicit inverse and for which both the function and its inverse are single-valued and continuous. For diodes and transistors the vanishing slope in the reverse direction is usually handled by placing a small leakage conductance across each junction.

Broyden [55] has recently proposed a variation of the Newton–Raphson technique. The method incorporates two modifications. The norm of the vector $g(v)$, which must tend to zero as a solution is approached, is never allowed to increase. The method also avoids computation of the inverse Jacobian matrix at each iteration. Instead, an arbitrary approximation to the matrix is chosen at the first iteration and then successively updated. Branin and Wang [56] have applied the method to nonlinear dc problems, including statistical analysis. More recently, Broyden [57] has concluded that enforced norm reduction is not always advantageous and that an adaption of his method used in conjunction with a particular form of Davidenko's method [58] may converge more rapidly. In addition, Brown [59] has proposed a variation of the Newton–Raphson technique in which an inequality constraint is applied to the norm of the vector $-J^{-1}(v)g(v)$.

Three termination criteria are commonly used. The first is to stop iterating when the absolute difference between each unknown voltage or current iterate and its previous value is reduced below some preset minimum. The second is to stop when the relative error, defined as the absolute difference divided by the value of the iterate, is reduced to a preset minimum. Finally, an approach sometimes used with nodal analysis is to require that the sum of the currents at each node be reduced to a preset minimum. All of the approaches have advantages and disadvantages in specific cases and none is superior in general.

Nonlinear DC Analysis Programs

Both NET-1 and its predecessor TAP formulate the dc equations separately from the transient situation. Further, both programs consider the linear and nonlinear portions of a network separately for dc analysis. The linear portion is analyzed using a modification of Kron's method of tearing [60]. Nonlinearities are treated as a set of side constraints, and Newton–Raphson iteration is used.

In SCEPTRE different trees are chosen for dc and transient analyses. A modified Newton–Raphson procedure [61] is used which ensures that the true operating point on an exponential is approached from below. This amounts to updating with current whenever junction voltage increases.

Both CIRCUS, using the state-variable approach, and TRAC, using nodal analysis, employ Newton–Raphson iteration while updating with voltage in the third quadrant and updating with current in the first quadrant. BIAS-3, using nodal analysis and Newton–Raphson iteration, updates with current in the first quadrant when a junction voltage increases and with voltage when it decreases. In the third quadrant BIAS-3 always updates with voltage while modeling the exponential characteristic in terms of a line through the origin rather than a tangent. This does not affect the dc solution and yet insures that if a junction voltage becomes positive, the new trial linearization point will lie in the first quadrant.

TRAC, CIRCUS, and BIAS-3 converge reasonably well on most moderately sized circuits of up to 40 nodes. Nonlinear models built into TRAC include junction diodes and bipolar transistors. The program's input format is cumbersome while, as supplied by the Harry Diamond Laboratory, TRAC includes several assembly language subroutines which may require translation. CIRCUS has the advantage of a free format input language, zener-diode, tunnel-diode, unijunction, and junction field-effect transistor models in a stored library. It is, however, a much more complex program to implement and modify. BIAS-3 is relatively small but is limited to nonlinear dc analysis of bipolar transistor circuits.

SCEPTRE is by far the most flexible program in that models may be built by the user, nested, and recalled as necessary. The price of this flexibility is size and complexity as the program consists of over 15 000 statements. Its dc solution has also been known to suffer from poor convergence properties.

One recent program should also be mentioned. The DICAP portion of SYSCAP is an extremely general dc analysis program. It is large and is currently available only to users of Control Data Corporation's CYBERNET remote batch system.

V. Nonlinear Transient Analysis

The general procedure for the transient analysis of a nonlinear circuit is to evaluate the state of the circuit at a given point in time and to extrapolate ahead to a new time point.

The computation time required for such an analysis program is directly proportional to the number of time increments into which the analysis (simulation) time must be divided. The total simulation time is usually a multiple of

the largest time constant (smallest eigenvalue) associated with the circuit linearized at a time point. On the other hand, for a large class of programs including NET-1, CIRCUS, and SCEPTRE, the length of a time step h in order to retain solution accuracy is determined by the smallest time constant (largest eigenvalue) within the circuit. Physically interpreted, this implies that if high-frequency devices are used in a low-frequency application, the computation time for an adequate simulation may be excessive. To understand how this problem is alleviated in programs such as TRAC and CIRPAC, it is necessary to examine the numerical integration process and the associated problem of stability.

Before proceeding, however, the use of the term numerical integration should be clarified. Strictly speaking, integration is associated with finding the area under some known function. The nonlinear transient analysis problem is one of obtaining the solution to a set of first-order nonlinear differential equations of the form (5). For the nonlinear situation, (5) can be written

$$\dot{x}(t) = f(x(t)). \tag{38}$$

The notation indicating the dependence of $f(x(t))$ on $u(t)$ has been dropped for convenience. Some initial condition

$$x(t)|_{t=0} = x(0) = x_0 \tag{39}$$

must be specified *a priori* or found as the result of a dc analysis. Because of the similarity of the formulas used in obtaining a solution to (38) to numerical integration formulas, the term has come to be loosely applied to both numerical processes.

Explicit Integration

Suppose at some point in time that $\dot{x}(t)$ is approximated by

$$\dot{x}(t) = \frac{x(t+h) - x(t)}{h}. \tag{40}$$

If (41) is substituted into (38), the result can be rearranged into the form

$$x(t+h) = hf(x(t)) + x(t). \tag{41}$$

Here $x(t+h)$ is defined explicitly in terms of $x(t)$. The numerical integration procedure defined by (41) is known as the forward Euler method [16] and is illustrated in Fig. 4(a). If $\hat{x}(t)$ is the exact solution to (38), it is easy to visualize the gross errors which can result from choosing h too large. The similarity of the right-hand side of (41) to a truncated Taylor series expansion about t suggests that the error in $x(t+h)$ will be proportional to h^2 multiplied by the second derivative of $x(t)$ evaluated someplace between t and $t+h$.

The forward Euler method defined by (41) is so low in accuracy that it is seldom used. Nonetheless, it serves to illustrate the idea that while the exact solution to (38) must be continuous, the computed solution is at best a piecewise-linear approximation to the exact solution.

In Fig. 4(a) the difference between the exact solution and the computed solution suggests the plausibility of using the

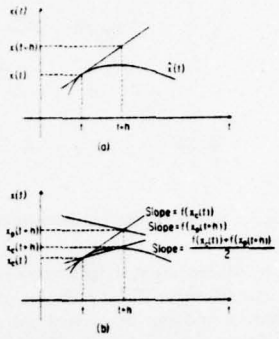


Fig. 4. Numerical integration. (a) Forward Euler method. (b) Improved predictor–corrector method.

average value of the derivatives at t and $t+h$ in (41). This results in what is called the trapezoidal integration method

$$x(t+h) = x(t) + \frac{h}{2}\left[f(x(t)) + f(x(t+h))\right]. \tag{42}$$

Equation (41) can be used to provide an initial approximation to $x(t+h)$ and thus $f(x(t+h))$. This is made clearer if (41) and (42) are rewritten as follows:

$$x_p(t+h) = hf(x_c(t)) + x_c(t) \tag{43}$$

$$x_c(t+h) = x_c(t) + \frac{h}{2}\left[f(x_c(t)) + f(x_p(t+h))\right]. \tag{44}$$

Equations (43) and (44) constitute a predictor–corrector pair, the predictor equation (43) providing $x_p(t+h)$ as an explicit function of $x_c(t)$ and the corrector equation (44) providing $x_c(t+h)$ as an explicit function of $x_c(t)$ and $x_p(t+h)$. This method is illustrated in Fig. 4(b). The increased accuracy which can result in relation to the forward Euler method is apparent as is the possibility of using more complex predictor–corrector pairs [62].

Implicit Integration

Consider now an alternative approach. The approximation to the derivative $\dot{x}(t)$ could just as easily have been made at $t+h$. In this case (41) becomes

$$x(t+h) = hf(x(t+h)) + x(t). \tag{45}$$

This is known as the backward Euler integration formula. Since $x(t)$ is known, (45) represents an implicit equation in the unknown $x(t+h)$. This equation may be solved for $x(t+h)$ by the Newton–Raphson method previously considered and the process repeated at each point in time. This procedure is called implicit numerical integration. The trapezoidal integration formula (42) is also an implicit integration formula.

All of the methods discussed thus far are of the general form

$$x(t+h) = \sum_{i=0}^{k} a_i x(t-ih) + h \sum_{j=-1}^{l} b_j \dot{x}(t-jh). \tag{46}$$

It is convenient to introduce an alternate notation. Without loss of generality it can be assumed that the time step h is constant and thus that $t=nh$. The notational change is defined such that

$$x(t+h) = x((n+1)h) = x_{n+1}. \tag{47}$$

Equation (46) can now be written as (48)

$$x_{n+1} = \sum_{i=0}^{k} a_i x_{n-i} + h \sum_{j=-1}^{l} b_j \dot{x}_{n-j}. \tag{48}$$

Methods used in which $b_{-1}=0$ such as (41) or (43) and (44) are termed explicit integration methods while methods used in which $b_{-1}\neq 0$ such as (42) or (45) are termed implicit integration methods.

Runge–Kutta Integration

A third class[2] of integration methods differs from the preceding two classes in that the interval between t and $t+h$ is divided into subintervals. These routines, which are referred to as Runge–Kutta methods [45], seek to establish a very good approximation to an average derivative in the subinterval. A fourth-order Runge–Kutta procedure is given by

$$x(t+h) = x(t) + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4) \tag{49}$$

where

$$f_1 = f(x(t), t) \tag{50a}$$

$$f_2 = f\left(x(t) + \frac{h}{2} f_1, t + \frac{h}{2}\right) \tag{50b}$$

$$f_3 = f\left(x(t) + \frac{h}{2} f_2, t + \frac{h}{2}\right) \tag{50c}$$

$$f_4 = f\left(x(t) + h f_3, t + \frac{h}{2}\right). \tag{50d}$$

In addition, both higher and lower order Runge–Kutta methods are available.

Stability of Numerical Integration

With the preceding material as background, the problem of step-size determination with regard to stability can be considered [15]. In this context, stability refers not to the response of the physical circuit but rather to whether or not the errors generated at each step of the numerical integration process tend to decay (stable) or grow (unstable) as the solution progresses in time.

The study of stability is carried out by substituting the differential equation of interest into the particular form of the integration formula being considered and generating a difference equation. The roots of the difference equation are found by converting the difference equation into an algebraic equation. (In some cases the z transform can be used.) If the order of the difference equation exceeds the order of the original differential equation, here assumed to be first order, then all but one of the solutions to the difference equation are parasitic. Stability requires that all parasitic solutions have magnitudes less than unity. Those parasitic solutions whose magnitudes exceed unity represent growing solution modes which can be excited by local truncation errors and thus dominate the correct solution.

Through a transformation, the solutions to the difference equation can be related to the eigenvalues of the circuit at each time point and the time step h. For explicit integration and Runge–Kutta methods,[3] the time step h must inevitably be held smaller than a constant divided by the largest eigenvalue at the present time point in order to keep the parasitic solutions small. The constant involved is usually less than ten [43]. On the other hand, for implicit methods, it is found that this requirement may be relaxed by three or more orders of magnitude [63]. As brought out earlier, an iterative method such as Newton–Raphson must be used to determine the state of the circuit at $t+h$. The additional computation time required to solve the nonlinear system of equations at a time point is more than offset by the comparatively large steps in simulation time which may be taken between points. Note that in this regard both the backward Euler and the trapezoidal methods are found to be stable for all positive values of h when the eigenvalues lie in the open left half-plane. This does not mean that circuits such as oscillators and multivibrators whose eigenvalues may lie in the right half-plane cannot be analyzed but rather that the maximum value which h may be allowed to assume must be reduced.

Truncation Error

Stability of a numerical integration method implies only that parasitic solutions when excited will not grow with time. Thus stability only guarantees that as time is allowed to go infinity, the computed solution will converge to the exact solution. At finite times the solutions may differ significantly due to truncation error. The error term associated with the forward Euler method mentioned previously is an example. In that particular case, the error is associated with h^2 multiplied by the second derivative or curvature of $x(t)$. Thus in regions where the response is rapidly varying, h may have to be chosen many times smaller than stability considerations require. The time step h can be increased when the response is slowly changing.

In general, when a higher order integration formula is used, the truncation error may be made proportional to higher derivatives. Similar to a Taylor series, higher order terms tend to zero. The order of an integration formula can be determined in the following way. If linear differential equations whose solutions are polynomials of finite order are considered, the order of the polynomial of largest degree

[2] An alternative classification scheme is based on the number of previous time points at which values of $x(t)$ or $\dot{x}(t)$ are required to compute $x(t+h)$. Single-step methods require values only at times greater than or equal to t. Multistep methods may require values at $t-h$, $t-2h$, etc. In this sense, the Runge–Kutta method described here is a single-step method. It is also possible to use Runge–Kutta methods in a multistep manner by considering subintervals of width h in a total interval of width nh where $n>1$.

[3] Here only second or higher order methods are implied where order is defined in the subsection on truncation error. For first-order methods, no parasitic solutions exist.

for which (48) is exact is the order of the method. Without further justification, it is stated that backward Euler is first order while trapezoidal integration is second order.

A difficulty with higher order methods is that the number of parasitic solutions to the stability difference equation is increased with the order, and the restrictions on h become increasingly severe. A compromise must be made between using highly stable methods with larger truncation errors and less stable methods with reduced truncation errors. An approach used by several writers [64]–[66] is to vary the order of the method as the calculation proceeds, based on an examination of the eigenvalues or, equivalently, the rate at which the response is varying. The aim, of course, is to use as high an order method as possible consistent with stability [64].

Nonlinear Transient Analysis Programs

With the exception of the Runge–Kutta approach, the numerical integration formulas considered thus far and generalized in terms of (48) have all been linear in the sense that the value x_{n+1} is expressed as a linear combination of function values and derivatives. Pope [67] and Fowler and Warten [68] have developed modifications of predictor–corrector methods in which an exponential term is included. This approach allows a larger step size to be used in comparison with straight predictor–corrector or Runge–Kutta methods and has been used in both SCEPTRE and CIRCUS. The approach does not allow the dramatic increase in time steps which is possible with an implicit method. SCEPTRE includes two additional integration routines, a trapezoidal predictor–corrector method similar to (43) and (44) and a fourth order Runge–Kutta method similar to (49) and (50).

CIRPAC, though not generally available outside of Bell Telephone Laboratories, demonstrates the advantages of implicit integration methods. The program uses the second-order implicit integration formula [43]

$$x_{n+1} = -\tfrac{1}{3}x_{n-1} + \tfrac{4}{3}x_n + \tfrac{2}{3}h\dot{x}_{n+1} \tag{51}$$

which is given here without justification. The step size h, which is allowed to vary, is kept as large as possible consistent with maintaining a small local truncation error. Shichman [43] reports that CIRPAC typically runs up to ten times faster than CIRCUS and up to twenty times faster than an earlier version of CIRPAC which used a predictor–corrector routine.

The programs considered to this point use state-variable formulations where the presence of a first-order differential equation is made readily apparent. The TRAC program which employs nodal analysis uses a trapezoidal implicit integration method. The formulation can be conveniently illustrated for the case of a linear capacitor where the i–v characteristic is given by

$$i_c = C\frac{dv_c}{dt}. \tag{52}$$

In integral form, (52) can be rewritten as

$$\int_t^{t+h} i_c(t)\,dt = C\int_t^{t+h} d\big[v_i(t) - v_j(t)\big] \tag{53}$$

where $v_c(t)$ is taken to be the difference between the node voltages $v_i(t)$ and $v_j(t)$. The trapezoidal integration formula (42) applied to the left-hand side of (53) yields

$$\frac{h}{2}\big[i_c(t+h) + i_c(t)\big] = C\big[v_i(t+h) - v_i(t) - v_j(t+h) + v_j(t)\big]. \tag{54}$$

This equation can be rewritten to obtain

$$i_c(t+h) = \frac{2C}{h}\big[v_i(t+h) - v_j(t+h)\big] - \frac{2C}{h}\big[v_i(t) - v_j(t)\big] - i_c(t). \tag{55}$$

This last expression can be represented in terms of an equivalent conductance $2C/h$ in parallel with a current source between nodes i and j of value $-(2C/h)\big[v_i(t)-v_j(t)\big] - i_c(t)$. Both elements are readily handled in terms of nodal analysis.

Other reactive circuit elements are handled by TRAC in a similar manner. The modified Newton–Raphson method used for dc analysis and previously described is used at each time point. While nearly as fast as CIRPAC, TRAC does not incorporate the same sort of variable step-size feature and truncation error control. Rather the user has the option of specifying up to ten time intervals and the maximum step size to be allowed in each time interval. Thus, where it is known that a switching transient is about to occur, the user can force a smaller step size to be imposed. An advantage of TRAC is that it is readily available. Further, it is a relatively small program which consists of only 2000 statements.

Recently, a modified version of TRAC known as TRACAP and released as a part of the SYSCAP package has been announced. Like DICAP its dc counterpart, TRACAP is only available to users of Control Data Corporation's remote batch service CYBERNET. A version of TRAC known as MTRAC which handles magnetic cores has also been developed [27].

The transient portion of ECAP is restricted to circuits in which nonlinear elements are described by fixed piecewise-linear models. Ideal switches are used to move from one linear segment of a model to another in accordance with the direction of current through a sensing branch. The use of this portion of the program is quite cumbersome if piecewise-linear reactive elements are included. Implicit integration is used.

As to the future, NET-II[4] a completely revised version of NET-I (in FORTRAN) is under extensive test.

VI. Linear Pole-Zero Analysis

Linear pole-zero circuit analysis is considered as a separate topic because of the number of different techniques used and the specialized problems involved relative to frequency-domain analysis. Naturally, the poles and zeros of the transfer function, once obtained, can provide the same frequency response information as the linear ac analysis programs previously considered. However, in many problems it is the poles and zeros themselves which are of prime interest to the circuit designer.

[4] NET-II is a new version of NET-I by A. Malmberg, now being developed.

State-Variable and Eigenvalue Approach

The CORNAP program written by Pottle has found widespread use. As mentioned previously it employs a state-variable formulation leading to a set of equations of the form of (3) and (4). In the complex frequency domain these equations take the form

$$sx = Ax + Bu \tag{56a}$$

$$y = Cx + Du. \tag{56b}$$

The output vector y becomes

$$y = [C(sI - A)^{-1}B + D]u. \tag{57}$$

For the single-input–single-output case, the circuit transfer function $T(s)$ is given by

$$T(s) = \frac{y(s)}{u(s)} = C(sI - A)^{-1}B + D \tag{58}$$

where D is a scalar. The poles of the transfer function (natural frequencies of the circuit) are given by the zeros of $\det(sI - A)$.

In an early version of CORNAP the recursive method of Leverrier [69], also known as the Souriau–Frame algorithm [70], was used to construct the adjoint of $(sI - A)$ and the characteristic polynomial associated with A. Muller's method [71] was then used to find the zeros of the polynomial. The recursive method however was particularly sensitive to roundoff errors which severely limited accuracy.

In the present version, use is made of the fact that the zeros of $\det(sI - A)$, which are the zeros of the characteristic polynomial and the poles of $T(s)$, are also the eigenvalues of the matrix A. The QR algorithm of Francis [72]–[74] is used to compute these eigenvalues.

The transmission zeros of a circuit transfer function are obtained by making use of the fact that the zeros of the transmission function of a feedback network placed around an ideal amplifier of infinite gain are the poles of the closed-loop transfer function. The transfer function $T(s)$ is considered to be placed around such an ideal amplifier. State equations describing this inverse system [75] are obtained in terms of the original A, B, C, and D matrices. The eigenvalues of the inverse system are then the zeros of the original system.

CORNAP can handle up to 32 state variables (eigenvalues) but does require a 64×64 double precision matrix plus an additional 32×64 double precision matrix. The core requirements are thus large and it has been noted that accuracy is questionable for some large circuits of the order of 64 branches and 24 nodes.

A recent paper by Sandberg and So [76] describes an alternate approach for obtaining transmission zeros. The approach still makes use of the QR algorithm for finding eigenvalues.

Iterative Technique

The TRFN portion of LISA uses a very accurate iterative procedure to find the natural frequencies of a circuit. A nodal formulation is used and each term of the nodal admittance matrix represents a polynomial in s, the complex frequency variable. For single input single output, the transfer function is

$$T_{mn}(s) = \frac{v_n(s)}{i_m(s)} = \frac{\det Y_{mn}(s)}{\det Y(s)}. \tag{59}$$

$Y_{mn}(s)$ is the minor of the element $y_{mn}(s)$, the poles of $T_{mn}(s)$ are the zeros of $\det Y(s)$. The zeros of $T_{mn}(s)$ are the zeros of $\det Y_{mn}(s)$. The iterative root-finding method of Muller is applied directly to $\det Y(s)$ and $\det Y_{mn}(s)$. In Muller's method, the determinant is initially evaluated at three points and modeled by a quadratic. A zero of this quadratic is then used in place of one of the original evaluation points. This procedure is continued and a sequence of better and better approximations to each zero of the determinant is developed. The amount of determinantal evaluation time appears excessive; however, the accuracy is excellent. Further, it is possible to program this approach very effectively to reduce greatly the computation time. FRANK,[5] a program of this type, is faster for large circuits than CORNAP, more accurate requiring only single precision, and uses significantly less core. Finally, where element values are to be varied and analyses repeated, a large savings in time may result because previous solutions can be used to supply good initial estimates of new solutions.

Laplace Expansion Approach

By a straightforward Laplace expansion of the determinant of the nodal admittance matrix $Y(s)$, the coefficients of the characteristic polynomial can be obtained. Muller's method can then be used to solve for the zeros of the polynomial. An approach of this type is used in the POLY portion of LISA; however, it is severely limited by the effects of roundoff error and is limited to circuits of less than 12 nodes.

Larger circuits can be handled effectively with programs of this type if a restriction is made to active RC circuits. Several polynomial manipulations are then eliminated and accuracy and speed are improved. Program SPRAGUE[6] has been found to be accurate and fast for circuits up to 19 nodes.

Nodal Analysis–Eigenvalue Approach

The eigenvalue approach based on a nodal equation formulation can be used for circuits restricted to active RC elements. The nodal admittance matrix has the form

$$Y(s) = G + sC. \tag{60}$$

Eigenvalue programs of this type have been developed both at M.I.T. [30], [31] and at the Technical University of Denmark [77]. The M.I.T. program requires the user to enter the elements of the matrices G and C of (60) directly; however, a topological input routine can easily be added.

[5] FRANK is a program written and developed in 1969 by the IC Group, Electronics Research Laboratory, University of California, Berkeley.

[6] SPRAGUE is a program written and developed in 1968 by the IC Group, Electronics Research Laboratory, University of California, Berkeley.

The method is based on the transformation of det $(G+sC)$ to det $(\hat{G}\hat{C}^{-1}-sI)$ where I is the identity matrix and $\hat{G}$ and $\hat{C}^{-1}$ are matrices of rank less than or equal to n the order of $Y(s)$. The transformation is carried out such that the zeros of the original determinant are unchanged. The zeros of the alternate determinant, however, are also the eigenvalues of $\hat{G}\hat{C}^{-1}$. The QR method is used to compute these eigenvalues. The transmission zeros are obtained by applying the same procedure to $Y_{mn}(s)$.

If the rank of $\hat{C}$ is equal to n, $\hat{G}\hat{C}^{-1}$ can be obtained by applying a variation of Gaussian elimination known as Gauss–Jordan reduction to C and performing the same operation on G. Gauss–Jordan reduction differs from Gaussian elimination in that a matrix is diagonalized and then normalized to obtain the identity matrix. This requires elimination of elements above the diagonal as well as those below. If the rank of $\hat{C}$ is less than n, then redundant rows and columns exist in $\hat{G}$ and $\hat{C}$ and must be eliminated [77].

This method again requires only sufficient core to store the nodal admittance matrix and is very fast. However, it is a relatively new method and its relative accuracy has yet to be determined, though it should be better than that of the Laplace expansion approach.

An extension to active *RLC* networks has also been described [78]. Inductive elements are replaced by ideal gyrator–capacitor equivalent circuits. The effectiveness of this extension to the preceding method has not yet been evaluated, however.

Summary

A number of techniques which are used for computing the poles and zeros of circuit transfer functions have been considered. Though each approach requires more extensive computations than are required to compute responses in the frequency domain, the additional insight obtained from a knowledge of the poles and zeros often merits the additional effort. In all cases, the order of the circuits which can be analyzed is significantly less than that which can be handled with the frequency-domain program described earlier.

No mention has been made of programs using a topological tree enumeration formulation. While accurate [79]–[80] and useful for sensitivity studies because of the explicit form of the coefficients of the polynomials, these programs have been found to be too slow to be of interest and are limited to very small circuits. Similar conclusions have been found to apply to programs based on flowgraph methods [81].

VII. Conclusion

In the preceding sections of this paper, the basic elements of computer-aided circuit analysis have been reviewed with emphasis on those techniques and routines necessary for the adequate simulation of four basic classes of circuits: linear dc and ac, nonlinear dc, nonlinear transient, and linear pole zero. One topic not considered but very important is the development of device models suitable for computer-aided circuit analysis. An overview of modeling would, in itself, require a full paper and thus cannot be undertaken here. However, several articles on the modeling of bipolar and field-effect transistor models are included in [82]–[85].

References

[1] F. F. Kuo and J. F. Kaiser, Eds., *Systems Analysis by Digital Computer*. New York: Wiley, 1966.

[2] G. J. Herskowitz, Ed., *Computer-Aided Integrated Circuit Design*. New York: McGraw-Hill, 1968.

[3] F. F. Kuo and W. G. Magnuson, Jr., Ed., *Computer Oriented Circuit Design*. Englewood Cliffs, N. J.: Prentice-Hall, 1969.

[4] H. Falk, "Computer programs for circuit design," *Electro-Technol.* (New York), vol. 77, pp. 54–57, June 1966.

[5] D. Christiansen, "Computer-aided design—I: the man-machine merger," *Electronics*, vol. 39, pp. 110–123, September 19, 1966.

[6] J. Dunaman, "Check design program availability," *Electron. Des.*, vol. 14, pp. 76–80, October 11, 1966.

[7] R. H. Dickhaut, "Comparison of three computer-aided design programs," *Electro-Technol.*, vol. 78, pp. 88–89, January 1967.

[8] ——, "Computer-aided design—VI: comparing the 'big two' programs," *Electronics*, vol. 40, pp. 74–90, February 1967.

[9] W. G. Magnuson, Jr., "Computer-aided design—VIII: picking transient analysis programs," *Electronics*, vol. 40, pp. 84–87, April 17, 1967.

[10] G. K. Pritchard, "A survey of transient circuit analysis programs," *1967 NASA Computer-Aided Circuit Design Seminar Proc.*, pp. 97–104.

[11] D. F. Dawson, F. F. Kuo, and W. G. Magnuson, Jr., "Computer-aided design of electronic circuits—a user's viewpoint," *Proc. IEEE*, vol. 55, pp. 1946–1954, November 1967.

[12] F. H. Branin, Jr., "Machine analysis of networks and its applications," IBM Development Lab., Poughkeepsie, N. Y., Tech. Rep. TR 00.855, 1962.

[13] ——, "Computer-aided design—IV: analyzing circuits by the numbers," *Electronics*, vol. 40, pp. 88–103, January 9, 1967.

[14] *Proc. IEEE* (Special Issue on Computer-Aided Design), vol. 55, November 1967.

[15] D. A. Calahan, *Computer-Aided Network Design*, preliminary ed. New York: McGraw-Hill, 1968.

[16] L. P. Huelsman, *Digital Computations in Basic Circuit Theory*. New York: McGraw-Hill, 1968.

[17] Cornell Electrical Engineering Conf., Biennial Conf. Proc. on Computerized Electronics, August 1969.

[18] T. R. Bashkow, "The A matrix, new network description," *Trans. IRE Circuit Theory*, vol. CT-4, pp. 117–119, September 1957.

[19] N. G. Brooks and H. S. Long, "A program for computing the transient response of transistor switching circuits—PETAP," IBM Development Lab., Poughkeepsie, N. Y., Tech. Rep. TR 00.700, 1959.

[20] F. H. Branin, Jr., "D-C analysis portion of PETAP—a program for analyzing transistor switching circuits," IBM Development Lab., Poughkeepsie, N. Y., Tech. Rep. TR 00.701, 1960.

[21] "1620 electronic circuit analysis program [ECAP][1620-EE-02X] user's manual," IBM Application Program File H20-0170-1, 1965.

[22] A. F. Malmberg, F. L. Cornwell, and F. N. Hofer, "NET-1 network analysis program," Los Alamos Scientific Lab., Los Alamos, N. Mex., Rep. LA-3119, 7090/94 version, August 1964.

[23] "Automated digital computer program for determining responses of electronic systems to transient nuclear radiation (PREDICT)," IBM Space Guidance Center, Oswego, N. Y., IBM File 64-521-5, July 1964.

[24] L. D. Milliman, W. A. Massena, and R. H. Dickhaut, "CIRCUS—a digital computer program for transient analysis of electronic circuits—user's guide," Boeing Co., Seattle, Wash., Harry Diamond Lab. Rep. AD-346-1, January 1967.

[25] H. W. Mathers, S. R. Sedore, and J. R. Seuts, "Automated digital computer program for determining responses of electronic circuits to transient nuclear radiation (SCEPTRE)," IBM Space Guidance Center, Oswego, N. Y., IBM File 66-928-611, February 1967.

[26] E. D. Johnson, C. T. Kleiner, L. R. McMurray, E. L. Steele, and F. A. Vassallo, "Transient radiation analysis by computer program (TRAC)," Autonetics Div., North American Rockwell Corp., Anaheim, Calif., Tech. Rep. issued by Harry Diamond Labs., June 1968.

[27] D. Nitzan and J. R. Herndon, "MITRAC—a computer program for analysis of circuits including magnetic cores," Stanford Research Institute, Menlo Park, Calif., SRI Project 6408, Rep. 6, June 1969.

[28] "NARCSYSCAP, system of circuit analysis programs," Autonetics Div., North American Rockwell Corp., 1969.

[29] W. J. McCalla and W. G. Howard, "BIAS-3, a program for the nonlinear dc analysis of bipolar transistor circuits," *1970 Int. Solid State Circuits Conf. Dig.*

[30] P. E. Gray and C. L. Searle, "NATFREQS (FORTRAN)," in *Electronic Principles, Physics, Models, and Circuits*. New York: Wiley, 1969, Appendix C.

[31] W. H. Ohm, "Applications of computer-aided circuit design and analysis," S.B. thesis, Massachusetts Institute of Technology, Cambridge, June 1968.

[32] K. L. Deckert and E. T. Johnson, "User's guide for LISA," IBM, San Jose, Calif., 7094-IBM0001, August 1967.

[33] ——, "LISA—a program for linear systems analysis," IBM, San Jose, Calif., presented at WESCON, Los Angeles, Calif., August 1966.

[34] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory*. New York: McGraw-Hill, 1969, ch. 12.

[35] E. S. Kuh and R. A. Rohrer, "The state-variable approach to network analysis," *Proc. IEEE*, vol. 53, pp. 672–686, July 1965.

[36] P. R. Bryant, "The order of complexity of electrical networks," *Proc. Inst. Elec. Eng.*, monograph 335E, vol. 106C, pp. 174–188, June 1959.

[37] ——, "The explicit form of Bashkow's *A* matrix," *Trans. IRE Circuit Theory* (Correspondence), vol. CT-9, pp. 303–306, September 1962.

[38] R. L. Wilson and W. A. Massena, "An extension of Bryant–Bashkow *A* matrix," *IEEE Trans. Circuit Theory* (Correspondence), vol. CT-12, pp. 120–122, March 1965.

[39] C. Pottle, "State-space techniques for general active network analysis," in *Systems Analysis by Digital Computer*, F. F. Kuo and J. F. Kaiser, Eds. New York: Wiley, 1966, ch. 3.

[40] ——, "A 'textbook' computerized state-space network analysis algorithm," *IEEE Trans. Circuit Theory* (Correspondence), vol. CT-16, pp. 566–568, November 1969.

[41] C. A. George, "BELAC user's manual," General Electric Co., Utica, N. Y., Tech. Info. Ser. R69EML11, August 1969.

[42] H. Shichman, "Computation of dc solutions for bipolar transistor networks," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 460–466, November 1969.

[43] ——, "The integration system of a nonlinear network-analysis program," *IEEE Trans. Circuit Theory*, vol. CT-17, pp. 378–386, August 1970.

[44] E. Isaacson and H. B. Keller, *Analysis of Numerical Methods*. New York: Wiley, 1966.

[45] A. Ralston, *A First Course in Numerical Analysis*. New York: McGraw-Hill, 1965.

[46] N. Sato and W. F. Tinney, "Techniques for exploiting the sparsity of the network admittance matrix," *IEEE Trans.* (Power App. Syst.), vol. 82, pp. 944–950, December 1963.

[47] W. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proc. IEEE*, vol. 55, pp. 1801–1809, November 1967.

[48] *Sparse Matrix Symp. Proc.*, IBM Rep. RA-1, March 1969.

[49] R. D. Berry, "An optimal ordering of electronic circuit equations for a sparse matrix solution," this issue, pp. 40–50.

[50] R. W. Jensen and M. D. Lieberman, *IBM Electronics Circuit Analysis Program: Techniques and Applications*. Englewood Cliffs, N. J.: Prentice-Hall, 1968.

[51] S. W. Director and R. A. Rohrer, "The generalized adjoint network and network sensitivities," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 318–323, August 1969.

[52] ——, "Automated network design—the frequency-domain case," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 330–337, August 1969.

[53] A. J. Broderson, S. W. Director, and W. A. Bristol, "Simultaneous automated ac and dc design of linear integrated circuit amplifiers," this issue, pp. 50–58.

[54] B. A. Wooley, "The computer-aided design optimization of integrated broadband amplifiers," *1970 ISSCC Dig. Tech. Papers*, pp. 74–75.

[55] C. G. Broyden, "A class of methods for solving nonlinear simultaneous equations," *Math. Comp.*, vol. 19, pp. 577–593, October 1965.

[56] F. H. Branin, Jr., and H. H. Wang, "A fast reliable iteration method for dc analysis of nonlinear networks," *Proc. IEEE*, vol. 55, pp. 1819–1826, November 1967.

[57] C. G. Broyden, "A new method of solving nonlinear simultaneous equations," *Comput. J.*, vol. 12, pp. 94–99, February 1969.

[58] D. F. Davidenko, "On a new method of numerical solution of systems of nonlinear equations," *Dokl. Akad. Nauk SSSR*, vol. 88, pp. 601–602, 1953.

[59] G. C. Brown, "DC analysis of nonlinear networks," *Electron. Lett.*, vol. 5, pp. 374–375, August 1969.

[60] G. Kron, "A method of solving very large physical systems in easy stages," *Proc. IRE*, vol. 42, pp. 680–686, April 1954.

[61] R. H. Dickhaut, "Advances in computer techniques for radiation effects calculations," Boeing Doc. D2-90571, 1964.

[62] R. W. Hamming, *Numerical Methods for Scientists and Engineers*. New York: McGraw-Hill, 1962.

[63] I. W. Sandberg and H. Shichman, "Numerical integration on systems of stiff nonlinear differential equations," *Bell Syst. Tech. J.*, vol. 47, pp. 511–527, April 1968.

[64] C. W. Gear, "Numerical integration of stiff ordinary differential equations," Department of Computer Science, University of Illinois, Urbana, Rep. 221, 1967.

[65] W. Liniger and R. A. Willoughby, "Efficient numerical integration of stiff systems of ordinary differential equations," IBM Watson Research Center, Yorktown Heights, N. Y., Res. Rep. RC-1970, 1968; *SIAM J. Numer. Anal.*, vol. 7, pp. 47–66, March 1970.

[66] P. M. Russo, "On the time domain analysis of linear time-invariant networks with large time-constant spreads by digital computer," this issue, pp. 194–197.

[67] D. A. Pope, "An exponential method of numerical integration of ordinary differential equations," *Comm. ACM*, vol. 6, pp. 491–493, August 1963.

[68] M. E. Fowler and R. M. Warten, "A numerical technique for ordinary differential equations with widely separated eigenvalues," *IBM J. Res. Develop.*, pp. 537–543, September 1967.

[69] A. S. Householder, *The Theory of Matrices in Numerical Analysis*. Waltham, Mass.: Blaisdell, 1964, pp. 166–168.

[70] L. A. Zadeh and C. A. Desoer, *Linear System Theory*. New York: McGraw-Hill, 1963, pp. 303–305.

[71] D. E. Muller, "A method for solving algebraic equations using an automated computer," in *Mathematical Tables and Other Aids to Computation*, vol. 10, pp. 208–215, 1956.

[72] J. G. F. Francis, "The Q–R transformation—I," *Comput. J.*, vol. 4, pp. 265–271, October, 1961; "The Q–R transformation—II," *Comput. J.*, vol. 4, pp. 332–345, January 1962.

[73] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*. New York: Oxford, 1965, ch. 8.

[74] B. N. Parlett, "The *LU* and *QR* algorithms," in *Mathematical Methods for Digital Computers*, vol. 2, A. Ralston and H. S. Wilf, Eds. New York: Wiley, 1967, ch. 5, pp. 116–130.

[75] R. W. Brockett, "Poles, zeros, and feedback: state space interpretation," *IEEE Trans. Automatic Control*, vol. AC-10, pp. 129–135, April 1965.

[76] I. W. Sandberg and H. C. So, "A two-sets-of-eigenvalues approach to the computer analysis of linear systems," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 509–517, November 1969.

[77] E. V. Sorensen, "Circuit analysis by algebraic eigenvalue technique," to be published.

[78] R. E. Parkin, "A state variable method of circuit analysis based on a nodal approach," *Bell Syst. Tech. J.*, pp. 1957–1970, November 1968.

[79] D. A. Calahan, "Linear network analysis and realization digital computer programs, and instruction manual," *University of Illinois Bulletin*, vol. 62, February 1965.

[80] E. V. Sorensen, "A preliminary note on the analytical network program ANPI," Laboratory of Circuit Theory, Technical University of Denmark, Lyngby, Rep. LKT23, 1967.

[81] L. P. McNamee and N. Potash, "A user's and programmer's manual for NASAP," University of California, Los Angeles, Calif., Rep. 68-38, August 1968.

[82] D. A. Hodges and H. Shichman, "Modeling and simulation of insulated-gate field-effect transistor switching circuits," *IEEE J. Solid-State Circuits*, vol. SC-3, pp. 285–289, September 1968.

[83] D. Frohman-Bentchkowsky and L. Vadasz, "Computer-aided design and characterization of digital MOS integrated circuits," *IEEE J. Solid-State Circuits*, vol. SC-4, pp. 57–64, April 1969.

[84] H. K. Gummel and H. C. Poon, "An integral charge control model of bipolar transistors" (to be published).

[85] F. A. Lindholm, "Integrated-circuit transistor and diode models for network-analysis programs," in this issue, pp. 122–128.

[86] R. D. Berry, private communication.

APPENDIX II

Reprinted from the IEEE Transactions on Circuit Theory, Vol. CT-18, No. 1, January 1971.

An Optimal Ordering of Electronic Circuit Equations for a Sparse Matrix Solution

ROBERT D. BERRY

Abstract—Sparse matrix storage and solution techniques are used extensively in solving very large systems of hundreds of linear equations which arise in the analysis of multiply interconnected physical systems. These techniques have often been overlooked in the analysis of relatively small electric networks even though their use can result in very significant improvements in computer storage requirements and execution times. The time savings is particularly noticeable when many solutions for the same circuit with different parameter values are required.

A particular sparse matrix storage, reordering, and solution technique is described. A node renumbering algorithm which is specifically directed at preserving the sparse structure of nodal admittance matrices during the solution by Gaussian elimination is described in detail. Computer flow charts for the renumbering are included along with specific circuit examples which compare the relative computational effort required for sparse solution versus full matrix solution.

I. Introduction

MANY circuit analysis programs that are in widespread use today are limited by the necessary storage requirements of the admittance or impedance matrix and are inefficient in obtaining the solution to the set of equations describing the network. If nodal analysis is used, the resulting admittance matrix of the circuit is virtually always sparse; that is, less than 50 percent of the matrix elements are nonzero.

Many authors have written on the subject of solutions of sparse systems of equations [1]–[3]. Results have indicated that time savings of orders of magnitude can be achieved for solution of large networks. This paper describes a solution algorithm which was specifically developed for nodal analysis of electric circuits. The algorithm takes advantage of the sparse nature of typical network admittance matrices to obtain the voltage solution vector by Gaussian elimination. Storage is allocated for only nonzero terms of the admittance matrix and Gaussian elimination proceeds through only these terms, thereby allowing efficient analysis of much larger circuits than was previously possible.

A FORTRAN IV program utilizing the sparse storage and solution algorithms has been written and is currently in use in a linear ac analysis program. It can handle 100 node circuits in less than 30 000 words of storage on the campus CDC 6400 computer.[1] The 29-node circuit shown in Appen-

Manuscript received June 17, 1970; revised August 10, 1970.

The author is with the Naval Weapons Center, China Lake, Calif. 93555.

[1] This program was developed by graduate students in the Department of Engineering, University of California, Berkeley, during 1969–1970.

dix II required just 3.5 s to obtain 71 complete solutions at a like number of frequency points. The time includes the reloading of the admittance matrix for each of the 71 frequency points.

The paper is divided into three sections and has Appendixes. Section II describes the system of pointers developed for rapid access of the nonzero elements of any network. Section III describes the decomposition and optimum renumbering scheme to preserve the sparse matrix structure during Gaussian elimination [4] on the admittance matrix. Section IV describes the decomposition and solution algorithms which operate only upon the nonzero terms of the matrix. Appendix I includes the flow charts for the renumbering scheme. Appendix II illustrates two circuit examples.

II. Admittance Matrix Storage

The admittance matrix Y for a typical ten-node electric circuit contains fewer than 50-percent nonzero terms. As the number of circuit nodes increases the percentage of nonzero terms drops, so that at 100 nodes typical circuits contain 5-percent nonzero terms. For efficient utilization of high-speed memory and to allow for practical solution of very large circuits, storage is allocated for only the nonzero terms of the admittance matrix. These terms are collapsed into the columnar array A. An efficient set of pointers for locating these terms in the array is an absolute necessity.

The matrix is symmetric except for terms associated with solid-state device equivalent circuits. For the most part, these terms are structurally symmetric but numerically nonsymmetric. Pointers locate nonzero terms and therefore can take advantage of structural symmetry, even though numeric symmetry does not exist.

The columnar array A of nonzero terms is organized into the three sections shown in the following table.

The first section of the A array is reserved for the diagonal terms of the admittance matrix (100 allowed). Position k contains the y_{kk} diagonal term; therefore, no pointer system is necessary.	$A(1)=y_{11}$ $A(2)=y_{22}$ $A(3)=y_{33}$ . . . $A(n)=y_{nn}$ . . .
The second section of the A array is reserved for the nonzero off-diagonal terms of the upper triangular portion of the admittance matrix (400 total allowed). These terms are stored by rows.	$A(100)$ $A(101)=y_{1j}$ $A(102)=y_{1k}$ . . . $A(m)=y_{n-1,n}$ . . .
The third section of the A array is reserved for the nonzero off-diagonal terms of the lower triangular portion of the admittance matrix (400 total allowed). These terms are stored by columns.	$A(500)$ $A(501)=y_{j1}$ $A(502)=y_{k1}$ . . . $A(m+400)=y_{n,n-1}$ . . . $A(900)$

The pointer system for these last two sections takes advantage of the symmetric structure, so that one set of pointers serves both of the triangular portions.

The pointers are set up in two stages. In the first stage, the nonzero structure of the nodal admittance matrix is recorded by the pointer system as the individual network elements and their respective node connections are transferred from data cards to the program. Based upon this structure, a node renumbering is effected which attempts to minimize the number of operations required for triangular decomposition of Y by Gaussian elimination. After completing the renumbering, the pointer system for the lower and upper triangular portion of Y is changed to correspond to the new node numbers. Storage of admittance terms in the second and third sections of the columnar array A is to be based upon the renumbered system, while the diagonal section is to be arranged according to the original node numbers. Only the pointers associated with the upper triangular array of the renumbered system of equations are retained. Numbers of the lower triangular portion can be located by using symmetry and the upper pointers.

The pointers consist of two integer arrays. The first array IUR contains N integer numbers where N is the number of circuit nodes. The number stored in position k of this array represents the starting location in the second pointer array IUC of terms associated with row k of the admittance matrix. In stage one the second array includes the column location of all nonzero off-diagonal terms in each row of the original Y matrix. In stage two the positions in the first pointer array and all of the numbers of the second pointer array correspond to the new node numbers. Every nonzero term that will occur in the final decomposed form of the upper triangular matrix is identified by this final pointer set. At this stage the position in the second pointer array directly corresponds to the position in the collapsed admittance array of the term identified. As an example to illustrate the pointer system, consider an admittance matrix with the following pattern of nonzero terms before node renumbering, where the y_{ij} are the nonzero terms:

$$\begin{bmatrix} y_{11} & 0 & y_{13} & 0 & y_{15} \\ 0 & y_{22} & y_{23} & y_{24} & 0 \\ y_{31} & y_{32} & y_{33} & y_{34} & 0 \\ 0 & y_{42} & y_{43} & y_{44} & 0 \\ y_{51} & 0 & 0 & 0 & y_{55} \end{bmatrix}.$$

The pointer arrays before node renumbering would contain the following numbers.

Row Locator	Column Identifier	Term Identified
IUR(1) = 1	IUC(1) = 3	y_{13}
IUR(2) = 3	IUC(2) = 5	y_{15}
IUR(3) = 5	IUC(3) = 3	y_{23}
IUR(4) = 8	IUC(4) = 4	y_{24}
IUR(5) = 10	IUC(5) = 1	y_{31}
IUR(6) = 11	IUC(6) = 2	y_{32}
	IUC(7) = 4	y_{34}
	IUC(8) = 2	y_{42}
	IUC(9) = 3	y_{43}
	IUC(10) = 1	y_{51}

After node renumbering the admittance matrix might assume the following form:

$$\begin{bmatrix} y_{11} & 0 & 0 & y_{14} & y_{15} \\ 0 & y_{22} & y_{23} & y_{24} & 0 \\ 0 & y_{32} & y_{33} & y_{34} & 0 \\ y_{41} & y_{42} & y_{43} & y_{44} & y_{45} \\ y_{51} & 0 & 0 & y_{54} & y_{55} \end{bmatrix}.$$

The numbers within the pointer arrays would be the following with the admittance terms in the A array as shown on the right.

IUR(1) = 1	IUC(1) = 4	$A(101) = y_{14}$
IUR(2) = 3	IUC(2) = 5	$A(102) = y_{15}$
IUR(3) = 5	IUC(3) = 3	$A(103) = y_{23}$
IUR(4) = 6	IUC(4) = 4	$A(104) = y_{24}$
IUR(5) = 7	IUC(5) = 4	$A(105) = y_{34}$
	IUC(6) = 5	$A(106) = y_{45}$
		.
		.
		.
		$A(501) = y_{41}$
		$A(502) = y_{51}$
		$A(503) = y_{32}$
		$A(504) = y_{42}$
		$A(505) = y_{43}$
		$A(506) = y_{54}$

The first pointer array IUR(k) indicates the starting location in IUC(j) of the nonzero terms identified in row (column) k of the admittance matrix.

The second pointer array IUC(j) indicates the column location of the nonzero term stored in $A(100+j)$ and the row location of the nonzero term stored in $A(500+j)$.

III. Decomposition and Node Renumbering

The first step toward obtaining the solution vector x in the set of equations $Yx = b$ involves triangular decomposition of Y, the symmetrically structured admittance matrix, by Gaussian elimination. This technique involves the breaking down of Y into a product of two unique matrices, L and U, where $Y = LU$. The matrix L is lower triangular in form with 1 as the value of every diagonal element,

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix}.$$

The matrix U is an upper triangular matrix:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}.$$

A full matrix decomposition will be illustrated to show the steps involved in a triangular decomposition by Gaussian elimination. The product of the L and U matrices shown is the following:

$$\underset{L}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix}} \underset{U}{\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix}} = \underset{Y}{\begin{bmatrix} y_{11} & y_{12} & y_{13} & y_{14} \\ y_{21} & y_{22} & y_{23} & y_{24} \\ y_{31} & y_{32} & y_{33} & y_{34} \\ y_{41} & y_{42} & y_{43} & y_{44} \end{bmatrix}}.$$

By examination

$$u_{11} = y_{11}$$
$$u_{12} = y_{12}$$
$$u_{13} = y_{13}$$
$$u_{14} = y_{14}.$$

By multiplying the rows of L by the first column of U, we see that

$$l_{21}u_{11} = y_{21} \text{ or } l_{21} = y_{21}/u_{11}$$
$$l_{31}u_{11} = y_{31} \text{ or } l_{31} = y_{31}/u_{11}$$
$$l_{41}u_{11} = y_{41} \text{ or } l_{41} = y_{41}/u_{11}.$$

The remaining u_{ij} and l_{ij} values in terms of y_{ij} and previously determined l_{ik}, u_{kj}, and u_{kk} values are

$$l_{21}u_{12} + u_{22} = y_{22} \text{ or } u_{22} = y_{22} - l_{21}u_{12}$$
$$l_{21}u_{13} + u_{23} = y_{23} \text{ or } u_{23} = y_{23} - l_{21}u_{13}$$
$$l_{21}u_{14} + u_{24} = y_{24} \text{ or } u_{24} = y_{24} - l_{21}u_{14}$$
$$l_{31}u_{12} + l_{32}u_{22} = y_{32} \text{ or } l_{32} = \frac{y_{32} - l_{31}u_{12}}{u_{22}}$$
$$l_{41}u_{12} + l_{42}u_{22} = y_{42} \text{ or } l_{42} = \frac{y_{42} - l_{41}u_{12}}{u_{22}}$$
$$l_{31}u_{13} + l_{32}u_{23} + u_{33} = y_{33} \text{ or } u_{33} = y_{33} - l_{31}u_{13} - l_{32}u_{23}$$
$$l_{31}u_{14} + l_{32}u_{24} + u_{34} = y_{34} \text{ or } u_{34} = y_{34} - l_{31}u_{14} - l_{32}u_{24}$$
$$l_{41}u_{13} + l_{42}u_{23} + l_{43}u_{33} = y_{43} \text{ or } l_{43} = \frac{y_{43} - l_{41}u_{13} - l_{42}u_{23}}{u_{33}}$$
$$l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + u_{44} = y_{44} \text{ or } u_{44} = y_{44} - l_{41}u_{14} - l_{42}u_{24} - l_{43}u_{34}.$$

The first row of U and the first column of L are obtained quite easily. In programming it is very desirable to replace the values of the y_{ij} with the values of either the u_{ij} or the l_{ij} as they are determined. Also, as each l_{i1} is determined the remaining values of y_{ik} in the ith row for $k>1$ are changed by subtracting from them the product $l_{i1}u_{1k}$. After this first row and first column step of the decomposition, the values in the Y storage locations will be the following:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & y_{22}^{(1)} & y_{23}^{(1)} & y_{24}^{(1)} \\ l_{31} & y_{32}^{(1)} & y_{33}^{(1)} & y_{34}^{(1)} \\ l_{41} & y_{42}^{(1)} & y_{43}^{(1)} & y_{44}^{(1)} \end{bmatrix}$$

where

$$\begin{aligned} y_{22}^{(1)} &= y_{22} - l_{21}u_{12} = u_{22} \\ y_{23}^{(1)} &= y_{23} - l_{21}u_{13} = u_{23} \\ y_{24}^{(1)} &= y_{24} - l_{21}u_{14} = u_{24} \\ y_{32}^{(1)} &= y_{32} - l_{31}u_{12} \\ y_{33}^{(1)} &= y_{33} - l_{31}u_{13} \\ y_{34}^{(1)} &= y_{34} - l_{31}u_{14} \\ y_{42}^{(1)} &= y_{42} - l_{41}u_{12} \\ y_{43}^{(1)} &= y_{43} - l_{41}u_{13} \\ y_{44}^{(1)} &= y_{44} - l_{41}u_{14}. \end{aligned}$$

Note that the values $y_{22}^{(1)}$, $y_{23}^{(1)}$, and $y_{24}^{(1)}$ are equal to u_{22}, u_{23}, and u_{24}, respectively. l_{32} and l_{42} are obtained from $y_{32}^{(1)}$ and $y_{42}^{(1)}$, respectively, by simply dividing by the pivot element u_{22}:

$$l_{32} = \frac{y_{32}^{(1)}}{u_{22}}$$

$$l_{42} = \frac{y_{42}^{(1)}}{u_{22}}.$$

Just as in the first row–column decomposition step, after each l_{i2} is obtained the remaining values of u_{ik} in the ith row for $k>2$ are changed by subtracting the product $l_{i2}u_{2k}$ from them. The values now found in the Y storage array are the following:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ l_{21} & u_{22} & u_{23} & u_{24} \\ l_{31} & l_{32} & y_{33}^{(2)} & y_{34}^{(2)} \\ l_{41} & l_{42} & y_{43}^{(2)} & y_{44}^{(2)} \end{bmatrix}$$

where

$$\begin{aligned} y_{33}^{(2)} &= y_{33}^{(1)} - l_{32}u_{23} \\ y_{34}^{(2)} &= y_{34}^{(1)} - l_{32}u_{24} \\ y_{43}^{(2)} &= y_{43}^{(1)} - l_{42}u_{23} \\ y_{44}^{(2)} &= y_{44}^{(1)} - l_{42}u_{24}. \end{aligned}$$

The decomposition proceeds by row–column steps until all the elements of U and L are determined and replace the elements of Y in storage. In general, at the ith row–column step, the ith row values of U are determined from the equation

$$u_{ij} = y_{ij}^{(i-1)}, \qquad \text{for all } j \geq i$$

and the ith column values of L are determined from the equation

$$l_{ji} = \frac{y_{ji}^{(i-1)}}{u_{ii}}, \qquad \text{for all } j > i.$$

This technique is associated with Doolittle [4].

The sparse matrix decomposition uses the Doolittle method but only carries out the operations that result in a finite change in an element value stored in the Y array. A finite change in a stored value occurs under just two circumstances. The first arises when solving for the l_{ji} term just mentioned and occurs whenever the $y_{ji}^{(i-1)}$ term is nonzero. (The diagonal term u_{jj} is always finite in the admittance matrix). The second circumstance arises just after the determination of a finite l_{ji} term when all of the $y_{jk}^{(i-1)}$ terms for $k>i$ are changed to $y_{jk}^{(i)}$ by the relationship

$$y_{jk}^{(i)} = y_{jk}^{(i-1)} - l_{ji}u_{ik}.$$

Once l_{ji} is determined to be nonzero, a change between $y_{jk}^{(i)}$ and $y_{jk}^{(i-1)}$ will occur only when u_{ik} is also nonzero. A node renumbering scheme is effected which attempts to organize the nonzero terms of the original admittance matrix in such a fashion as to cause the second circumstance to arise only when $y_{jk}^{(i-1)}$ is already nonzero. Otherwise an additional nonzero entry will occur in the matrix and will increase the total computational effort required for the decomposition.

There are three basic parts to the renumbering algorithm. All parts search the nonzero structure recorded by the stage one pointer system described previously. The algorithm takes advantage of structural symmetry whenever possible to speed up the renumbering process. An array NUMOFF is set up which records the total number of nonzero off-diagonal terms associated with each node. NUMOFF(k) equals the total number of these terms that would appear in the Y array in row k.

Part I of the algorithm searches this array once to see if there are any nodes with only one nonzero off-diagonal term. If one is found, it is numbered 1 and the array NUMOFF is altered. The off-diagonal term of this new node 1 will be located in some column j. Because of the symmetric structure, there will also be an off-diagonal term in row j, column 1. During the course of the decomposition, these two mentioned off-diagonal terms will become u_{1j} and l_{j1}, respectively. The only $y_{jk}^{(1)}$ term that will be altered during the first row–column step of the decomposition will be the y_{jj} term, which will change to

$$y_{jj}^{(1)} = y_{jj} - l_{j1}u_{1j}.$$

All of the other u_{1k} and l_{k1} for $k>1$ terms are zero.

Since all diagonal terms, i.e., all y_{jj}, are always nonzero to start with, there will be no additional nonzero terms created at this step of the decomposition. In searching for a node to number 2, we can remove from consideration both of the nonzero terms u_{1j} and l_{j1}. Neither of these terms will be used again during the course of the decomposition. To simulate their removal the array NUMOFF is changed by reducing the recorded number of off-diagonal terms associated with node j by 1. If, by this reduction of 1, the effective number of off-diagonal terms associated with node j is now one or fewer, then the jth node will be numbered 2 and the process repeated. A single sweep through the array NUMOFF will rapidly pick off every node that has only one or fewer effective off-diagonal terms. Decomposition of these single off-diagonal term equations will cause no new nonzero terms in the matrix.

A typical example showing the working array after the first row–column decomposition step is shown in the following. Row one has just one nonzero off-diagonal term. The array NUMOFF contains the effective number of nonzero off-diagonal terms shown on the right after 1 is subtracted from row 5:

$$\begin{bmatrix} u_{11} & 0 & 0 & 0 & u_{15} & 0 \\ 0 & y_{22} & 0 & y_{24} & 0 & y_{26} \\ 0 & 0 & y_{33} & 0 & y_{35} & 0 \\ 0 & y_{42} & 0 & y_{44} & y_{45} & 0 \\ l_{51} & 0 & y_{53} & y_{54} & y_{55}^{(1)} & 0 \\ 0 & y_{62} & 0 & 0 & 0 & y_{66} \end{bmatrix} \quad \begin{aligned} &\text{NUMOFF}(1) = 1 \\ &\text{NUMOFF}(2) = 2 \\ &\text{NUMOFF}(3) = 1 \\ &\text{NUMOFF}(4) = 2 \\ &\text{NUMOFF}(5) = 2 \\ &\text{NUMOFF}(6) = 1. \end{aligned}$$

Part II of the algorithm searches the remaining nodes (those not renumbered in Part I) for nodes which can be decomposed without increasing the number of nonzero terms. Suppose $m-1$ nodes were renumbered by the Part I search and suppose node i has associated with it two nonzero off-diagonal terms (NUMOFF(i)=2) in the portion of the matrix that has not been renumbered yet. Let these terms be y_{ij} and y_{ik}. Should we choose to renumber this node as node m, the decomposed terms would become u_{mj} and u_{mk}. By symmetry there would also be a calculation of the finite terms l_{jm} and l_{km}. Two terms in row j and two terms in row k would be altered by the decomposition. These terms are

$$\begin{aligned} y_{jj}^{(m)} &= y_{jj}^{(m-1)} - l_{jm}u_{mj} \\ y_{jk}^{(m)} &= y_{jk}^{(m-1)} - l_{jm}u_{mk} \\ y_{kj}^{(m)} &= y_{kj}^{(m-1)} - l_{km}u_{mj} \\ y_{kk}^{(m)} &= y_{kk}^{(m-1)} - l_{km}u_{mk}. \end{aligned}$$

The first and fourth terms are on the diagonal and are already nonzero. The second and third terms are off-diagonal but are in symmetric locations. Therefore, only one of these terms need be checked to see if it is nonzero. If $y_{jk}^{(m-1)}$ should happen to be nonzero, then the node would be renumbered next and no change in the structure of the Y array would occur during the decomposition. The j and k row would have 1 removed from the effective number of off-diagonal terms, and if this caused the effective number to drop to 1 in either row j or k, then that particular row would be renumbered next.

As each node is checked, an array IFILL is set up which records the number of new positions that would become nonzero if that particular node were renumbered next.

After checking all of the nonrenumbered nodes for those which can be decomposed without causing an increase in the nonzero terms, a check is made to see if any were renumbered before the last entry into the Part II algorithm. If any nodes were renumbered, the algorithm is repeated because now the effective number of nonzero off-diagonal terms is different from the time Part II was first entered. It is still possible that additional nodes can be renumbered that will not increase the nonzero terms during decomposition. When a complete Part II search is made without finding any nodes for renumbering, then Part III is entered. At this point, every remaining node would cause a change in the nonzero structure if it were to be renumbered next. The array IFILL indicates how many new nonzero terms would be created during the decomposition if the node in question were to be renumbered next.

Part III finds the node that would cause the fewest new nonzero terms by searching the array IFILL. In case more than one node would cause the same fewest new nonzero terms, the node among these with the most number of effective nonzero off-diagonal terms recorded in NUMOFF is numbered next. The reason for this choice is that the number of new nonzero terms created during the decomposition at this step is still a minimum, but also the number of terms removed from the nonrenumbered portion is the largest possible, subject to the first constraint. By this choice the sum of the numbers in the array NUMOFF is minimized.

After the choice is made and a node renumbered, the new nonzero topology caused by decomposition of that nodal equation is recorded in the system of pointers. The array NUMOFF is kept up to date by adding 1 to the row in which each new nonzero term caused by the decomposition of that node appears. Also, as in all prior renumbering in Part I and Part II, the array NUMOFF is altered by subtracting 1 from the appropriate rows containing the nonzero off-diagonal terms of the node just renumbered. If, by this subtraction, an effective number of 1 off-diagonal terms appears in any of the nonrenumbered rows, that row is immediately renumbered next. After the bookkeeping operations have been completed for renumbering of a row from Part III, Part II is entered at the beginning. The search proceeds from this point as if it were the first entry into II. Fig. 1 illustrates the matrix Y at an intermediate stage of renumbering. The X represent the locations of all nonzero terms of an admittance matrix for a 23-node circuit. The numbers around the immediate borders of the matrix represent the node numbers supplied by the designer. The order in which the

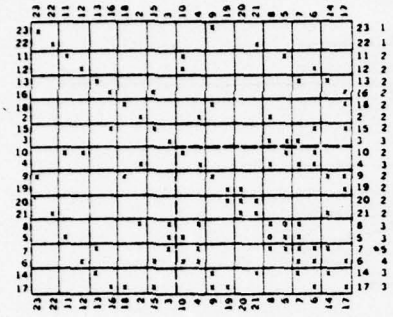


Fig. 1.

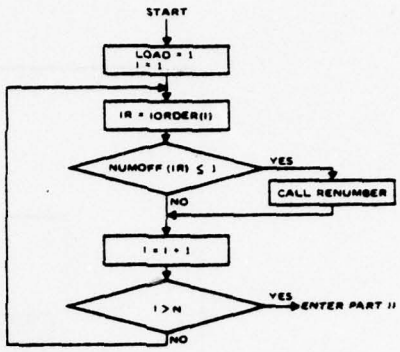


Fig. 2. Part I of the node renumbering algorithm.

numbers appear is the arrangement for decomposition at an intermediate stage of the renumbering. The numbers at the far right are the current numbers in the array NUMOFF. Note that in the first ten rows these numbers indicate the number of nonzero off-diagonal terms in the upper triangular portion of the array. The renumbering is complete to this tenth row which lies just above the dashed horizontal line. The two dashed lines outline a square lower right corner array which is referred to as the nonrenumbered portion of the matrix. The extreme right-hand numbers opposite the rows in this smaller array indicate the number of nonzero off-diagonal terms that lie within this lower right corner array. If any nonrenumbered node was renumbered 11, it would have this number of terms in the upper triangular matrix U.

For the tenth row, node 3 was selected instead of node 10 because either one would cause two new nonzero terms during decomposition, but node 3 had more off-diagonal terms in the nonrenumbered portion. The two zeros at the intersections of nodes 8 and 5 indicate the two locations for these new nonzero terms.

After all of the nodes are renumbered into the order in which the nodal equations will appear in the admittance matrix, the stage one pointers are reorganized. In the reorganization all of the pointers are changed to correspond to the new system of equation numbers and include all nonzero terms that will ultimately be found in the upper triangular matrix U. Terms of the lower triangular matrix L are located by symmetry.

IV. Sparse Matrix Solution

The node voltage vector can now be obtained by direct solution using Gaussian elimination with the sparse matrix technique. Because of the renumbering, the nonzero terms of the admittance matrix are located in the columnar array in the order in which decomposition will proceed, by rows in the upper triangular portion and by columns in the lower. This storage arrangement was selected to allow for very efficient decomposition processing and completely eliminates the need for any further search for nonzero terms.

Once all the values of L and U are determined by the decomposition, the process of forward and back substitution is employed to obtain the solution vector x. The steps involved are the following:

$$Yx = b$$
$$LU = Y$$
$$Lz = b, \quad \text{i.e., } z = L^{-1}b$$
$$Ux = z, \quad \text{i.e., } x = U^{-1}L^{-1}b.$$

The L^{-1} and U^{-1} matrices are never really determined in solving for the z or x vectors just mentioned. Forward substitution is used to find z and back substitution is used to determine x. A slight modification to the standard forward substitution technique is convenient because the nonzero terms of the L matrix are stored by column. The modified forward substitution process proceeds by columns instead of by rows and results in the same number of operations as the row approach. Elements of the z vector replace elements of the b vector as they are determined. The following is an example of the modified forward substitution technique for a 4×4 full matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{43} & 1 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix}.$$

The first column step is

$$z_1 = b_1$$
$$z_2^{(1)} = b_2 - l_{21}z_1$$
$$z_3^{(1)} = b_3 - l_{31}z_1$$
$$z_4^{(1)} = b_4 - l_{41}z_1.$$

The second column step is

$$z_2 = z_2^{(1)}$$
$$z_3^{(2)} = z_3^{(1)} - l_{32}z_2$$
$$z_4^{(2)} = z_4^{(1)} - l_{42}z_2.$$

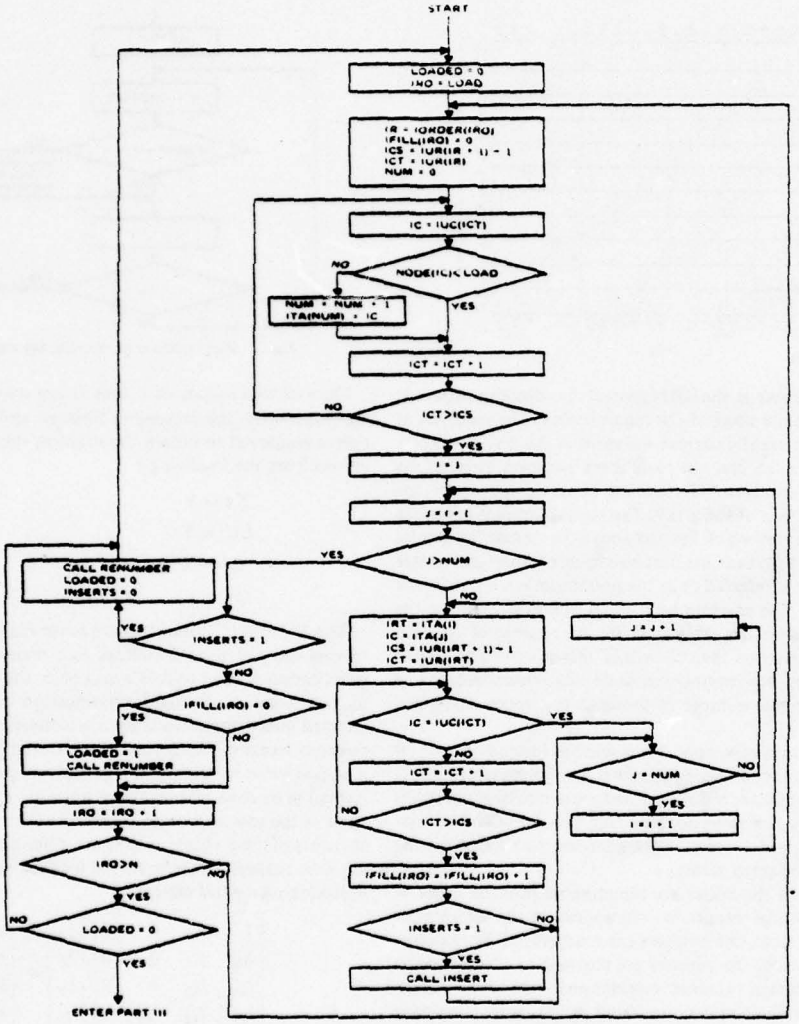


Fig. 3. Part II of the node renumbering algorithm.

The third column step is

$$z_3 = z_3^{(2)}$$
$$z_4^{(3)} = z_4^{(2)} - l_{43}z_3.$$

The fourth column step is

$$z_4 = z_4^{(3)}.$$

In general,

$$z_i^{(k)} = z_i^{(k-1)} - l_{ik}z_k.$$

Just as in the sparse matrix decomposition, only those operations that cause a finite change in a stored value in the b vector are carried out.

The back substitution used to determine the final solution vector x is a standard technique proceeding from the final nth row equation backward to the first row. The technique for a 4×4 full matrix example is illustrated as:

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & u_{14} \\ 0 & u_{22} & u_{23} & u_{24} \\ 0 & 0 & u_{33} & u_{34} \\ 0 & 0 & 0 & u_{44} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix}.$$

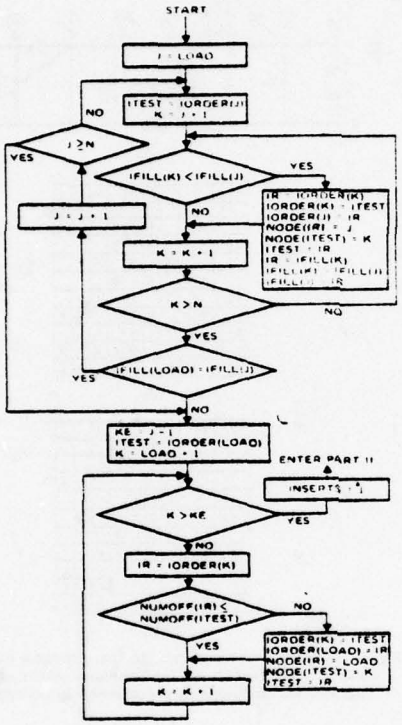


Fig. 4. Part III of the node renumbering algorithm.

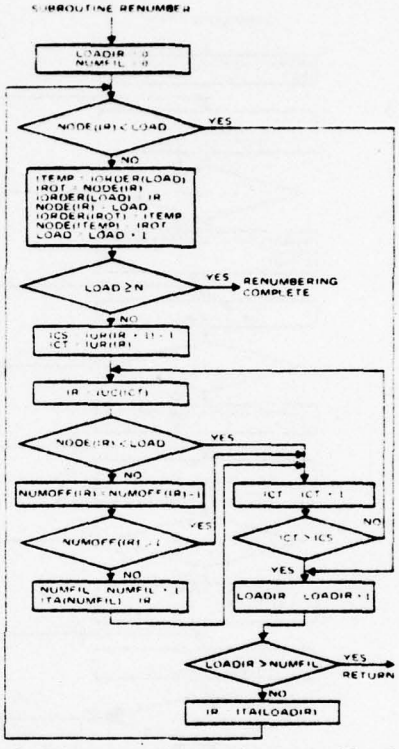


Fig. 5. Subroutine renumber called from Part I and Part II of the node renumbering algorithm.

Proceeding backward from the fourth equation we get the following answers:

$$x_4 = \frac{z_4}{u_{44}}$$

$$x_3 = \frac{z_3 - u_{34}x_4}{u_{33}}$$

$$x_2 = \frac{z_2 - u_{23}x_3 - u_{24}x_4}{u_{22}}$$

$$x_1 = \frac{z_1 - u_{12}x_2 - u_{13}x_3 - u_{14}x_4}{u_{11}}.$$

In general

$$x_i = \frac{z_i - \sum_{j=i+1}^{n} u_{ij}x_j}{u_{ii}}.$$

As in all previous sparse matrix modifications to the standard techniques, only those multiplication and addition operations involving nonzero quantities are carried out. The elements of x replace the elements of z in storage as they are determined.

V Summary

The time savings that have resulted from using the described sparse matrix techniques in lieu of a standard full matrix algorithm is very significant for obtaining a frequency response to even relatively small circuits in the 15- to 30-node category. The structure of nonzero terms and the node renumbering is determined just once for any given circuit. The most significant time savings result when many solutions for either different circuit parameter values or different excitation frequencies are required. An ideal application for the sparse matrix and renumbering algorithm arises in automated network design in which a search for optimum circuit parameter values is carried out in the frequency domain [5].

The standard triangular decomposition by Gaussian elimination requires on the order of $n^3/3$ multiplication-addition operations. The sparse matrix decomposition requires some number proportional to n operations but is very dependent upon the nonzero structure of a given circuit. Empirical results based upon the analysis of typical 20- to 30-node integrated circuit amplifiers has indicated this number varies from $4n$ to $16n$, but other circuit examples that would fall both above and below this range could be found.

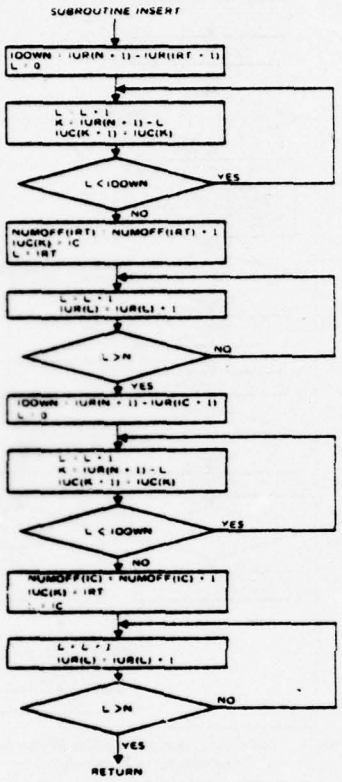


Fig. 6. Subroutine insert called from Part II of the node renumbering algorithm.

The forward-substitution and back-substitution operations are also proportional to n and dependent upon nonzero structure. Nearly one half of the total solution effort is expended in these last two algorithms. Additional time savings of up to a factor of 2 could be achieved under some circumstances, if the node renumbering also takes into account the sparse solution efficiency of the forward substitution and back substitution. Many times in a frequency response only one or two node voltages are desired. If these nodes are numbered last, the back substitution could be stopped in its first stages after these desired voltages are determined. Normally only one excitation source exists in the circuit. The b vector (current vector) consists of nearly all zero values. If the nodes associated with the nonzero b values were to be numbered next to last, the forward substitution could begin near the right-hand side of the L matrix and need only proceed through the final few nodes. Unfortunately, by arbitrarily numbering certain nodes last, the decomposition efficiency will most likely suffer. However, it has been shown experimentally that the total solution effort and the computer time will be reduced by one third to one half in most cases when this scheme is employed.

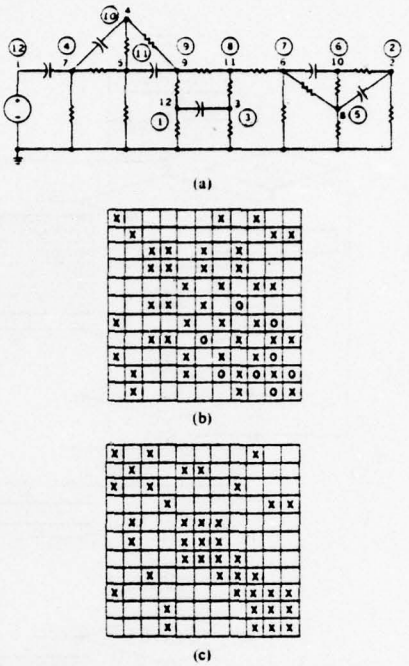
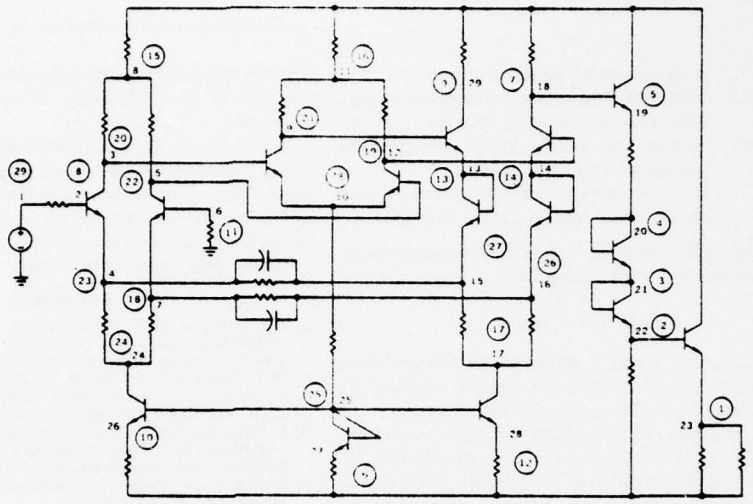


Fig. 7. Example 1. (a) The circuit. (b) The admittance matrix using the uncircled node numbers. (c) The admittance matrix using the circled node numbers determined by the renumbering algorithm.

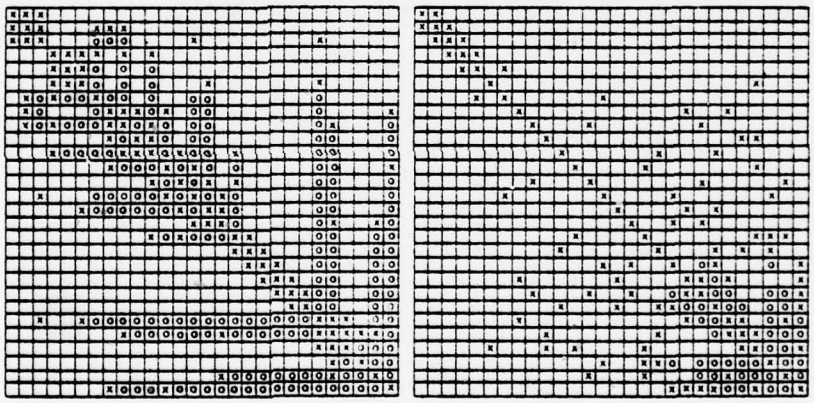
Appendix I

The node renumbering algorithm is broken down into the three parts described in Section III. Flow charts for these three parts are given in Figs. 2–4. The two subroutines called from these flow charts are shown in Figs. 5 and 6. The computer variables are defined as follows:

LOAD	equals the next node number to be assigned during renumbering; starts at 1 and proceeds sequentially to N.
N	number of nodes in the circuit excluding the reference node.
IORDER(j)	array containing the original node numbers in the order in which decomposition will proceed.
NODE(j)	array complementary to IORDER(j), i.e., if IORDER(j) = k, NODE(k) = j.
IUR(j)	array indicating the starting location of terms in IUC(k) associated with row j of the matrix.
IUC(j)	array indicating the column location of the term stored in $A(100+j)$ or the row location of the term stored in $A(500+j)$.
NUMOFF(j)	array indicating the effective number of nonzero off-diagonal terms left in row j at any simulated stage of the decomposition; the j is the original node number.



(a)



(b) (c)

Fig. 8 Example 2. (a) The circuit. (b) The admittance matrix using the uncircled node numbers. (c) The admittance matrix using the circled node numbers determined by the renumbering algorithm.

IFILL(j) array indicating the number of new nonzero terms that would be created in the L or U matrix if currently numbered row j was to be renumbered as row number LOAD.

INSERTS flag indicating whether or not insertion into the pointer arrays of a potential new nonzero term caused by decomposition should be carried out; if it equals 1 then the insertion is done, otherwise it is not.

all others dummy integer variables and integer arrays.

Appendix II

Figs. 7 and 8 are two circuit examples that compare the computational effort required for triangular decomposition by Gaussian elimination using a full matrix algorithm, the sparse matrix algorithm, and the sparse matrix algorithm with the node renumbering algorithm. In both circuits the reference node is numbered zero and all terms in the admittance matrix associated with the zero node are dropped. The resulting nonsingular matrix is partitioned as

$$\begin{bmatrix} Y_{nn} & Y_{ns} \\ Y_{sn} & Y_{ss} \end{bmatrix} \begin{bmatrix} v_n \\ v_s \end{bmatrix} = \begin{bmatrix} i_n \\ i_s - i_s \end{bmatrix}$$

where

Y_{nn} square array of nodal admittance terms corresponding to nodes of unknown voltage;

Y_{ss} square array of nodal admittance terms corresponding to nodes of grounded independent voltage sources;

Y_{ss} and Y_{sn} intersections of the above two sets;
v_n and i_n unknown voltage and independent current source vector associated with the Y_{nn} nodes;
v_s and i_s voltage source and current source vector associated with the Y_{ss} nodes;
i_x unknown current vector of currents through the grounded independent voltage source.

The node voltage vector v_n is to be determined by Gaussian elimination on the set of equations

$$Y_{nn}v_n = i_n - Y_{ns}v_s.$$

It is the Y_{nn} matrix that is shown in the two examples, Figs. 7 and 8. The X indicate the nonzero terms of Y_{nn} before decomposition. The zeros indicate terms which are zero in Y_{nn} but change to finite values during the decomposition into LU form. In example 1, Fig. 7, the standard full matrix decomposition requires 55 divisions, 385 multiplications, and 385 additions. By using the sparse matrix technique without renumbering, the decomposition requires 19 divisions, 41 multiplications, and 41 additions. With renumbering before sparse decomposition, these operations are reduced to 15 divisions, 25 multiplications, and 25 additions.

In example 2, Fig. 8, the full matrix decomposition requires 378 divisions, 6930 multiplications, and 6930 additions. This compares with 134 divisions, 746 multiplications, and 746 additions for sparse decomposition without renumbering and 63 divisions, 169 multiplications, and 169 additions with renumbering before sparse decomposition.

References

[1] E. C. Ogbuobiri, W. F. Tinney, and J. W. Walker. "Sparsity-directed decomposition for Gaussian elimination on matrices," 1969 Power Industry Computer Applications Conf. pp. 215–225.
[2] W. F. Tinney and J. W. Walker, "Direct solutions of sparse network equations by optimally ordered triangular factorization," *Proc. IEEE.*, vol. 55, pp. 1801–1809, November 1967.
[3] F. Gustavson, W. Liniger, and R. Willoughby, "Symbolic generation of an optimal Crout algorithm for sparse systems of linear equations," IBM Corp., Yorktown Heights, N. Y., Res. Paper RC 1852.
[4] L. Fox, *An Introduction to Numerical Linear Algebra.* New York: Oxford, 1965, pp. 60–65, 99–102.
[5] S. W. Director and R. A. Rohrer, "Automated network design—the frequency-domain case," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 330–337, August 1969.

APPENDIX III

STANFORD UNIVERSITY
DEPARTMENT OF ELECTRICAL ENGINEERING

J P FREFET
R W CUTTON

1 AUG 72

PROGRAM SPICE WAS DEVELOPED AT THE UNIVERSITY OF CALIFORNIA, BERKELEY. THE GENERAL CAPABILITIES OF SPICE ARE DESCRIBED BELOW AND IN THE AUGUST 1971 ISSUE OF THE IEEE JOURNAL OF SOLID STATE CIRCUITS. SPICE IS EQUIVALENT TO A THIRD GENERATION VERSION OF PROGRAM CANCER (DESCRIBED IN THE JOURNAL ARTICLE).

UNIVERSITY OF CALIFORNIA
COLLEGE OF ENGINEERING
DEPARTMENT OF ELECTRICAL ENGINEERING
AND COMPUTER SCIENCES

L W NAGEL
D O PEDERSON

8 MAY 72

USERS GUIDE FOR SPICE 1

SPICE IS A GENERAL PURPOSE CIRCUIT SIMULATION PROGRAM FOR NONLINEAR DC, NONLINEAR TRANSIENT, AND LINEAR AC ANALYSIS. CIRCUITS MAY CONTAIN RESISTORS, CAPACITORS, INDUCTORS, INDEPENDENT VOLTAGE AND CURRENT SOURCES, VOLTAGE DEPENDENT CURRENT SOURCES, AND THE FOUR MOST COMMON SEMICONDUCTOR DEVICES: BJTS, DIODES, JFETS AND MOSFETS.

SPICE HAS BUILT-IN MODELS FOR THE SEMICONDUCTOR DEVICES, AND THE USER SPECIFIES ONLY THE PERTINENT MODEL PARAMETER VALUES. TWO MODELS ARE AVAILABLE FOR THE BJT. THE SIMPLER MODEL IS BASED ON THE EBERS-MOLL MODEL AND INCLUDES CHARGE STORAGE EFFECTS, OHMIC RESISTANCES, AND A CURRENT DEPENDENT OUTPUT CONDUCTANCE. A MODEL BASED ON THE INTEGRAL CHARGE MODEL OF GUMMEL AND POON IS ALSO AVAILABLE FOR PROBLEMS WHICH REQUIRE A MORE SOPHISTICATED BJT MODEL. THE DIODE MODEL CAN BE USED FOR EITHER JUNCTION DIODES OR SHOTTKY BARRIER DIODES. THE JFET AND MOSFET MODELS ARE BOTH BASED ON THE FET MODEL OF SHICHMAN AND HODGES.

PROGRAM LIMITATIONS

200	NODES, INCLUDING INTERNAL DEVICE NODES. EACH NONZERO OHMIC RESISTANCE IN A DEVICE WILL GENERATE AN INTERNAL NODE. FOR EXAMPLE, A CIRCUIT WITH 35 USER SPECIFIED NODES AND 10 BJTS WITH NONZERO BASE AND COLLECTOR RESISTANCES WILL CONTAIN 55 NODES.
50	DEVICES (BJTS, DIODES, JFETS, AND MOSFETS).
25	INDEPENDENT VOLTAGE OR CURRENT SOURCES. ONLY 5 INDEPENDENT SOURCES CAN BE TIME DEPENDENT FOR TRANSIENT ANALYSIS.
200	TOTAL ELEMENTS, INCLUDING DEVICES AND INDEPENDENT SOURCES.
10	OUTPUT VARIABLES. AN OUTPUT VARIABLE IS EITHER A NODE TO NODE VOLTAGE OR A CURRENT THROUGH AN INDEPENDENT VOLTAGE SOURCE. OUTPUT VARIABLES MAY BE PRINTED IN TABULAR FORM, PLOTTED AS LINE PRINTER PLOTS, OR BOTH. ONLY 5 OUTPUT VARIABLES CAN BE USED IN THE AC SMALL SIGNAL ANALYSIS.
10	SETS OF MODEL PARAMETERS FOR DEVICES.

TYPES OF ANALYSIS

DC ANALYSIS

THE DC ANALYSIS PORTION OF SPICE DETERMINES THE DC OPERATING POINT OF THE CIRCUIT WITH INDUCTORS SHORTED AND CAPACITORS OPENED. A DC ANALYSIS IS AUTOMATICALLY PERFORMED PRIOR TO A TRANSIENT ANALYSIS TO DETERMINE THE TRANSIENT INITIAL CONDITIONS, AND PRIOR TO AN AC SMALL SIGNAL ANALYSIS TO DETERMINE THE LINEARIZED, SMALL SIGNAL MODELS FOR NONLINEAR DEVICES. IF REQUESTED, THE DC SMALL SIGNAL VALUE OF A TRANSFER FUNCTION (RATIO OF OUTPUT VARIABLE TO INPUT SOURCE), INPUT RESISTANCE, AND OUTPUT RESISTANCE WILL ALSO BE COMPUTED AS A PART OF THE SMALL SIGNAL OPERATING POINT. THE DC ANALYSIS CAN ALSO BE USED TO GENERATE DC TRANSFER CURVES. A SPECIFIED INDEPENDENT VOLTAGE OR CURRENT SOURCE IS STEPPED OVER A USER SPECIFIED RANGE AND THE DC OUTPUT VARIABLES ARE STORED FOR EACH SEQUENTIAL SOURCE VALUE. THE DC ANALYSIS OPTIONS ARE SPECIFIED ON THE .DC CONTROL CARD (PAGE 15).

AC SMALL SIGNAL ANALYSIS

THE AC SMALL SIGNAL PORTION OF SPICE COMPUTES THE AC OUTPUT VARIABLES AS A FUNCTION OF FREQUENCY. THE PROGRAM FIRST COMPUTES THE DC OPERATING POINT OF THE CIRCUIT AND DETERMINES LINEARIZED, SMALL SIGNAL MODELS FOR ALL OF THE NONLINEAR DEVICES IN THE CIRCUIT. THE RESULTANT LINEAR CIRCUIT IS THEN ANALYZED OVER A USER SPECIFIED RANGE OF FREQUENCIES. THE DESIRED OUTPUT OF AN AC SMALL SIGNAL ANALYSIS IS USUALLY A TRANSFER FUNCTION (VOLTAGE GAIN, TRANSIMPEDANCE, ETC). IF THE CIRCUIT HAS ONLY ONE AC INPUT, IT IS CONVENIENT TO SET THAT INPUT TO UNITY AND ZERO PHASE, SO THAT OUTPUT VARIABLES HAVE THE SAME VALUE AS THE TRANSFER FUNCTION OF THE OUTPUT VARIABLE WITH RESPECT TO THE INPUT.

THE GENERATION OF WHITE NOISE BY RESISTORS AND SEMICONDUCTOR DEVICES CAN ALSO BE SIMULATED WITH THE AC SMALL SIGNAL PORTION OF SPICE. EQUIVALENT NOISE SOURCE VALUES ARE DETERMINED AUTOMATICALLY FROM THE SMALL SIGNAL OPERATING POINT OF THE CIRCUIT, AND THE CONTRIBUTION OF EACH NOISE SOURCE IS ADDED AT A GIVEN SUMMING POINT. THE TOTAL OUTPUT NOISE LEVEL AND THE EQUIVALENT INPUT NOISE LEVEL ARE DETERMINED AT EACH FREQUENCY POINT. THE OUTPUT AND INPUT NOISE LEVELS ARE NORMALIZED WITH RESPECT TO THE SQUARE ROOT OF THE NOISE BANDWIDTH AND HAVE THE UNITS VOLTS/RT HZ OR AMPS/RT HZ. THE OUTPUT NOISE AND EQUIVALENT INPUT NOISE CAN BE PRINTED OR PLOTTED IN THE SAME FASHION AS OTHER OUTPUT VARIABLES.

THE FREQUENCY RANGE AND THE NOISE ANALYSIS OPTIONS ARE SPECIFIED ON THE .AC CONTROL CARD (PAGE 15).

TRANSIENT ANALYSIS

THE TRANSIENT ANALYSIS PORTION OF SPICE COMPUTES THE TRANSIENT OUTPUT VARIABLES AS A FUNCTION OF TIME OVER A USER SPECIFIED TIME INTERVAL. THE INITIAL CONDITIONS ARE AUTOMATICALLY DETERMINED BY A DC ANALYSIS. ALL SOURCES WHICH ARE NOT TIME DEPENDENT (FOR EXAMPLE, POWER SUPPLIES) ARE SET TO THEIR DC VALUE. FOR LARGE SIGNAL SINUSOIDAL SIMULATIONS, A FOURIER ANALYSIS OF THE OUTPUT WAVEFORM CAN BE SPECIFIED TO OBTAIN THE FREQUENCY DOMAIN FOURIER COEFFICIENTS. THE TRANSIENT TIME INTERVAL AND THE FOURIER ANALYSIS OPTIONS ARE SPECIFIED ON THE .TRAN CONTROL CARD (PAGE 16).

ANALYSIS AT DIFFERENT TEMPERATURES

ALL INPUT DATA FOR SPICE IS ASSUMED TO HAVE BEEN MEASURED AT 27 DEG C (300 DEG K). THE SIMULATION ALSO ASSUMES A NOMINAL TEMPERATURE OF 27 DEG C. THE CIRCUIT CAN BE SIMULATED AT UP TO 5 DIFFERENT TEMPERATURES BY USING A .TEMP CONTROL CARD (PAGE 15).

TEMPERATURE APPEARS EXPLICITLY IN THE EXPONENTIAL TERMS OF THE BJT AND DIODE MODEL EQUATIONS. IN ADDITION, SATURATION CURRENTS HAVE A BUILT-IN TEMPERATURE DEPENDENCE. THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT IN THE BJT MODELS IS DETERMINED BY:

```
IS (TEMP) = IO * (TEMP**3) * EXP (-Q * EG / (K * TEMP)),
```

WHERE K IS BOLTZMANS CONSTANT, Q IS THE ELECTRONIC CHARGE, IO IS A CONSTANT, AND EG IS THE ENERGY GAP WHICH IS A MODEL PARAMETER. THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT IN THE JUNCTION DIODE MODEL IS DETERMINED BY:

```
IS (TEMP) = IO * (TEMP**(3/N)) * EXP (-Q * EG / (K * TEMP)),
```

WHERE N IS THE EMISSION COEFFICIENT, WHICH IS A MODEL PARAMETER, AND THE OTHER SYMBOLS HAVE THE SAME MEANING AS ABOVE. FOR SHOTTKY BARRIER DIODES, THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT IS DETERMINED BY:

```
IS (TEMP) = IO * (TEMP**(2/N)) * EXP (-Q * EG / (K * TEMP)).
```

CONVERGENCE

BOTH DC AND TRANSIENT SOLUTIONS ARE OBTAINED BY AN ITERATIVE PROCESS WHICH IS TERMINATED WHEN THE NODE VOLTAGES CONVERGE TO WITHIN A TOLERANCE OF 0.1 PERCENT OR 50 MICROVOLTS, WHICHEVER IS LARGER. ALTHOUGH THE PARTICULAR ALGORITHM USED IS SPICE HAS BEEN FOUND TO BE VERY RELIABLE, IN SOME CASES IT WILL FAIL TO CONVERGE TO A SOLUTION. WHEN THIS HAPPENS, THE PROGRAM WILL PRINT OUT THE LAST NODE VOLTAGES AND TERMINATE THE JOB. THE NODE VOLTAGES THAT ARE PRINTED ARE NOT NECESSARILY CORRECT OR EVEN CLOSE TO THE CORRECT SOLUTION.

FAILURE TO CONVERGE IN THE DC ANALYSIS IS USUALLY DUE TO AN ERROR IN SPECIFYING CIRCUIT CONNECTIONS, ELEMENT VALUES, OR MODEL PARAMETER VALUES. REGENERATIVE SWITCHING CIRCUITS OR CIRCUITS WITH POSITIVE FEEDBACK PROBABLY WILL NOT CONVERGE IN THE DC ANALYSIS. FAILURE TO CONVERGE IN THE TRANSIENT ANALYSIS CAN ALSO BE DUE TO A TIME STEP WHICH IS TOO LARGE. SPICE PRESENTLY DOES NOT HAVE AN AUTOMATIC TIME STEP CONTROL, AND SIGNIFICANT ERROR AND/OR NONCONVERGENCE CAN RESULT IF THE TIME STEP IS LARGE COMPARED TO THE CIRCUIT TIME CONSTANTS.

INPUT FORMAT

THE INPUT FORMAT FOR SPICE IS OF THE FREE FORMAT TYPE. FIELDS ON A CARD ARE SEPARATED BY ONE OR MORE BLANKS, A COMMA, OR AN EQUAL (=) SIGN. SPACES PRECEDING OR FOLLOWING A COMMA OR EQUAL SIGN ARE IGNORED. A CARD MAY BE CONTINUED ONTO THE FOLLOWING CARD BY PUNCHING A + BEFORE THE FIRST FIELD ON THE CONTINUATION CARD.

A NAME FIELD MUST BEGIN WITH A LETTER (A THRU Z) AND CANNOT CONTAIN COMMAS OR BLANKS. ONLY THE FIRST SEVEN CHARACTERS OF THE NAME ARE USED.

A NUMBER FIELD MAY BE AN INTEGER FIELD (12,-44), A FLOATING POINT FIELD (3.14159), EITHER AN INTEGER OR A FLOATING POINT NUMBER FOLLOWED BY AN INTEGER EXPONENT (1E-14, 2.65E3), OR EITHER AN INTEGER OR A FLOATING POINT NUMBER FOLLOWED BY ONE OF THE FOLLOWING SCALE FACTORS:

G	1.0E9
MEG	1.0E6
K	1.0E3
M	1.0E-3
U	1.0E-6
N	1.0E-9
P	1.0E-12

LETTERS IMMEDIATELY FOLLOWING A NUMBER THAT ARE NOT SCALE FACTORS ARE IGNORED, AND LETTERS IMMEDIATELY FOLLOWING A SCALE FACTOR ARE IGNORED. HENCE, 10, 10V, 10VOLTS, AND 10HZ ALL REPRESENT THE SAME NUMBER, AND M, MA, MSEC, AND MMHOS ALL REPRESENT THE SAME SCALE FACTOR. NOTE THAT 1000, 1000.0, 1000HZ, 1E3, 1.0E3, 1KHZ, AND 1K ALL REPRESENT THE SAME NUMBER.

CIRCUIT DESCRIPTION

THE CIRCUIT TO BE ANALYZED IS DESCRIBED TO SPICE BY A SET OF ELEMENT CARDS, WHICH DEFINE THE CIRCUIT TOPOLOGY AND ELEMENT VALUES, AND A SET OF CONTROL CARDS, WHICH DEFINE THE MODEL PARAMETERS AND THE RUN CONTROLS. THE FIRST CARD IN THE INPUT DECK MUST BE A TITLE CARD, AND THE LAST CARD MUST BE A .END CARD. THE ORDER OF THE REMAINING CARDS IS ARBITRARY.

NODE NUMBERS MUST BE INTEGERS. THE DATUM NODE MUST BE NUMBERED 0 (ZERO). NODES NEED NOT BE NUMBERED SEQUENTIALLY. THE CIRCUIT CANNOT CONTAIN A LOOP OF VOLTAGE SOURCES AND/OR INDUCTORS AND CANNOT CONTAIN A CUTSET OF CURRENT SOURCES AND/OR CAPACITORS. EACH NODE IN THE CIRCUIT, INCLUDING THE DATUM NODE, MUST HAVE AT LEAST TWO CONNECTIONS.

ELEMENT CARDS

***** RESISTORS, CAPACITORS, INDUCTORS

GENERAL FORM
```
RXXXXXX N1 N2 VALUE
CXXXXXX N1 N2 VALUE
LXXXXXX N1 N2 VALUE
```

EXAMPLES
```
R13 12 17 1K
CGOOD 13 0 10P
LLINKS 42 69 1U
```

N1 AND N2 ARE THE TWO ELEMENT NODES. THE ORDER OF THE NODES FOR THESE ELEMENTS IS UNIMPORTANT. VALUE IS THE RESISTANCE (OHMS), THE CAPACITANCE (FARADS), AND THE INDUCTANCE (HENRIES), RESPECTIVELY. THIS VALUE CANNOT BE NEGATIVE OR ZERO.

***** VOLTAGE CONTROLLED CURRENT SOURCES

GENERAL FORM
```
IXXXXXX V N+ N- NC+ NC- VALUE DELAY
```

EXAMPLES
```
ISORS V 13 12 14 12 1.0M
IGM V 1 20 4 20 -2.0M 3.0NS
```

THE LETTER V MUST BE IN THE FIELD FOLLOWING THE ELEMENT NAME. N+ AND N- ARE THE POSITIVE AND NEGATIVE NODES, RESPECTIVELY. CURRENT FLOWS FROM THE POSITIVE NODE, THRU THE SOURCE, TO THE NEGATIVE NODE. NC+ AND NC- ARE THE POSITIVE AND NEGATIVE CONTROLLING NODES, RESPECTIVELY. VALUE IS THE TRANSCONDUCTANCE (MHOS).

IN THE AC ANALYSIS THE TRANSCONDUCTANCE CAN BE MODIFIED BY AN OPTIONAL DELAY (LINEAR PHASE) OPERATOR. THE DELAY (SECONDS) IS APPENDED AFTER THE VALUE. IF A DELAY, TD, IS INCLUDED, THE COMPLEX, FREQUENCY DEPENDENT VALUE OF TRANSCONDUCTANCE IS DETERMINED BY:

$$GM = VALUE * EXP(-J * 6.28318 * FREQ * TD) \quad .$$

THE DELAY IS IGNORED IN THE DC AND TRANSIENT ANALYSES.

***** INDEPENDENT SOURCES

```
GENERAL FORM        VXXXXXX N+ N- DC DCVAL AC ACVAL PHASE
                    IXXXXXX N+ N- DC DCVAL AC ACVAL PHASE

EXAMPLES            VCC 10 0 DC 6
                    IZENER 13 15 DC 600MA
                    VIN 13 2 DC 0.001 AC 1
                    IIN 21 23 AC 0.333 45.0
                    VMEAS 12 9
```

N+ IS THE POSITIVE NODE AND N- IS THE NEGATIVE NODE. NOTE THAT VOLTAGE SOURCES NEED NOT BE GROUNDED. CURRENT FLOWS FROM THE POSITIVE NODE, THRU THE SOURCE, TO THE NEGATIVE NODE.

DCVAL IS THE DC VALUE OF THE SOURCE. THE SOURCE IS SET TO THIS VALUE FOR DC ANALYSIS AND, IF NO TIME DEPENDENCE IS ATTACHED, IN THE TRANSIENT ANALYSIS. IF THE DC SOURCE VALUE IS ZERO, THE LETTERS DC AND THE DC VALUE CAN BE OMITTED.

ACVAL IS THE AC VALUE AND PHASE IS THE AC PHASE. THE SOURCE IS SET TO THIS VALUE IS THE AC ANALYSIS. THE ARBITRARY PHASE FACTOR CAN BE OMITTED. IF THE SOURCE IS NOT AN AC SMALL SIGNAL INPUT, THE LETTERS AC AND THE AC VALUES ARE OMITTED.

A SOURCE MAY BE GIVEN A TIME DEPENDENCE FOR THE TRANSIENT ANALYSIS BY APPENDING ONE OF THE THREE PREDEFINED FUNCTIONS: PULSE, EXPONENTIAL, AND SINUSOIDAL. IF PARAMETERS OTHER THAN SOURCE VALUES ARE OMITTED OR SET TO ZERO, THE DEFAULT VALUES SHOWN WILL BE ASSUMED. TSTEP IS THE PRINTING INCREMENT (TIME STEP), AND TSTOP IS THE FINAL TIME (PAGE 15).

1. PULSE — PULSE V1 V2 TD TR TF PW PER

```
EXAMPLE             VIN 3 0 PULSE -1 1 2NS 2NS 2NS 50NS 100NS
```

PARAMETERS AND DEFAULT VALUES

V1	INITIAL VALUE	---
V2	PULSED VALUE	---
TD	DELAY TIME	TSTEP
TR	RISE TIME	TSTEP
TF	FALL TIME	TSTEP
PW	WIDTH	TSTOP
PER	PERIOD	TSTOP

A SINGLE PULSE IS DESCRIBED BY THE FOLLOWING PIECEWISE LINEAR TABLE.

TIME	VALUE
0	V1
TD	V1
TD+TR	V2
TD+TR+PW	V2
TD+TR+PW+TF	V1
TSTOP	V1

2. EXPONENTIAL EXP V1 V2 TD1 TAU1 TD2 TAU2

EXAMPLE VIN 3 0 EXP -4 -1 2NS 30NS 60NS 40NS

PARAMETERS AND DEFAULT VALUES

V1	INITIAL VALUE	---
V2	PULSED VALUE	---
TD1	RISE DELAY TIME	TSTEP
TAU1	RISE TIME CONSTANT	TSTEP
TD2	FALL DELAY TIME	TSTEP
TAU2	FALL TIME CONSTANT	TSTEP

TIME	VALUE
0 TO TD1	V1
TD1 TO TD2	V1+(V2-V1)(1-EXP(-(T-TD1)/TAU1))
TD2 TO TSTOP	V1+(V2-V1)(1-EXP(-(T-TD1)/TAU1))+(V1-V2)(1-EXP(-(T-TD2)/TAU2))

3. SINUSOIDAL SIN VO VA FREQ TD THETA

EXAMPLE VIN 3 0 SIN 0 1 100MEG 1NS 1E10

PARAMETERS AND DEFAULT VALUES

VO	OFFSET	---
VA	AMPLITUDE	---
FREQ	FREQUENCY (IN HZ)	1/TSTOP
TD	DELAY	TSTEP
THETA	DAMPING FACTOR	0

TIME	VALUE
0 TO TD	VO
TD TO TSTOP	VO + VA * EXP(-(T-TD) * THETA) * SINE (6.28318 * FREQ * T)

SOURCES MAY BE GIVEN ANY COMBINATION OF VALUES (DC, AC, OR TRANSIENT), AND THESE VALUES MAY BE SPECIFIED IN ANY ORDER AS LONG AS THEY FOLLOW THE PROPER KEY WORD.

EXAMPLES
```
VIN 13 12 SIN 0 1 10MEG DC 0.1 AC 1 45
IZ 19 0 DC 0 PULSE 0 1 AC 0.5
VEQ 12 0 DC 0.5 EXP 0.5 0.9 10NS 40NS 70NS 40NS AC 1
```

***** BIPOLAR JUNCTION TRANSISTORS

GENERAL FORM Qxxxxxx NC NB NE MNAME AREA

EXAMPLE CAMP33 7 9 1 MOD1 2.0

NC IS THE COLLECTOR NODE, NB IS THE BASE NODE, NE IS THE EMITTER NODE, MNAME IS THE MODEL NAME (PAGE 9) AND AREA IS THE AREA FACTOR. THE AREA FACTOR IS EQUIVALENT TO THE NUMBER OF PARALLEL DEVICES. AN AREA FACTOR OF 2.0 IMPLIES THAT TWO TRANSISTORS OF THE SAME MODEL ARE CONNECTED IN PARALLEL. IF THE AREA IS OMITTED, AN AREA FACTOR OF 1.0 IS ASSUMED.

***** JUNCTION DIODES

GENERAL FORM Dxxxxxx N+ N- MNAME AREA

EXAMPLE DBRIDGE 8 10 DIODE1

N+ IS THE POSITIVE NODE, N- IS THE NEGATIVE NODE, MNAME IS THE MODEL NAME (PAGE 9), AND AREA IS THE AREA FACTOR (SEE BJTS, ABOVE).

***** JUNCTION FIELD EFFECT TRANSISTORS

GENERAL FORM Jxxxxxx ND NG NS MNAME AREA

EXAMPLE J1 7 2 3 JM1

ND IS THE DRAIN NODE, NG IS THE GATE NODE, NS IS THE SOURCE NODE, MNAME IS THE MODEL NAME(PAGE 9), AND AREA IS THE AREA FACTOR (SEE BJTS, ABOVE).

***** MOSFETS

GENERAL FORM Mxxxxxx ND NG NS NB MNAME AREA

EXAMPLE M31G 2 3 4 7 MLONG

ND IS THE DRAIN NODE, NG IS THE GATE NODE, NS IS THE SOURCE NODE, NB IS THE BULK (SUBSTRATE) NODE, MNAME IS THE MODEL NAME (PAGE 9), AND AREA IS THE AREA FACTOR (SEE BJTS, ABOVE).

***** .MODEL CARD

GENERAL FORM .MODEL MNAME TYPE PNAME1=PVAL1 PNAME2=PVAL2 ...

EXAMPLE .MODEL MOD1 NPN BF=50 IS=1E-13 VA=50

THE .MODEL CARD SPECIFIES A SET OF MODEL PARAMETERS THAT WILL BE USED BY ONE OR MORE DEVICES. MNAME IS THE MODEL NAME, AND TYPE IS ONE OF THE FOLLOWING TEN TYPES:

NPN	NPN EBERS-MOLL BJT MODEL
PNP	PNP EBERS-MOLL BJT MODEL
NGP	NPN GUMMEL-POON BJT MODEL
PGP	PNP GUMMEL-POON BJT MODEL
D	JUNCTION DIODE MODEL
SBD	SHOTTKY BARRIER DIODE MODEL
NJF	N CHANNEL JFET MODEL
PJF	P CHANNEL JFET MODEL
NMO	N CHANNEL MOSFET MODEL
PMO	P CHANNEL MOSFET MODEL

PARAMETER VALUES ARE DEFINED BY APPENDING THE PARAMETER NAME, AS GIVEN BELOW FOR EACH MODEL TYPE, FOLLOWED BY AN EQUAL SIGN AND THE PARAMETER VALUE. MODEL PARAMETERS THAT ARE NOT GIVEN A VALUE ARE ASSIGNED THE DEFAULT VALUE GIVEN BELOW FOR EACH MODEL TYPE.

MODEL PARAMETER VALUES CAN ALSO BE SPECIFIED AS A STRING OF NUMBERS IN THE ORDER GIVEN BELOW FOR EACH MODEL TYPE. THE FOLLOWING MODEL SPECIFICATION IS EQUIVALENT TO THE PREVIOUS MODEL CARD EXAMPLE:

EXAMPLE .MODEL MOD1 NPN 50,,,,,,,,,,1E-13,,,50

——— DIODE MODELS (BOTH JUNCTION AND SBD)

THE ONLY DIFFERENCE BETWEEN THE JUNCTION DIODE MODEL AND THE SHOTTKY BARRIER DIODE MODEL IS THE TEMPERATURE DEPENDENCE OF SATURATION CURRENT (SEE PAGE 3). THE DC CHARACTERISTICS OF THE DIODE ARE DETERMINED BY THE PARAMETERS IS AND N. AN OHMIC RESISTANCE, RS, IS INCLUDED. CHARGE STORAGE EFFECTS ARE MODELED BY A TRANSIT TIME, TT, AND A NONLINEAR DEPLETION LAYER CAPACITANCE WHICH VARIES AS THE -1/2 POWER OF JUNCTION VOLTAGE AND IS DEFINED BY THE PARAMETERS CJO AND PHI. THE ENERGY GAP, EG, AFFECTS ONLY THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT (SEE PAGE 3).

	NAME	PARAMETER	DEFAULT	TYPICAL
1	RS	OHMIC RESISTANCE	0	10
2	TT	TRANSIT TIME	0	0.1NS
3	CJO	ZERO BIAS JUNCTION CAPACITANCE	0	2PF
4	IS	SATURATION CURRENT	1.0E-14	1.0E-14
5	N	EMISSION COEFFICIENT	1	1.0
6	PHI	JUNCTION POTENTIAL	1	0.6
7	EG	ENERGY GAP	1.11 SI	1.11 FOR SI
			0.69 SBD	0.69 FOR SBD
				0.67 FOR GE

----- EBERS-MOLL BJT MODELS (BOTH NPN AND PNP)

THE EBERS-MOLL BJT MODEL USES THE DC EBERS-MOLL MODEL AS A BASIS. THE DC CHARACTERISTICS OF THE DEVICE ARE DETERMINED BY THE PARAMETERS BF AND BR, THE FORWARD AND REVERSE CURRENT GAINS, VA, WHICH DETERMINES THE OUTPUT CONDUCTANCE, AND THE SATURATION CURRENT, IS. THREE OHMIC RESISTANCES, RB, RC, AND RE, HAVE BEEN INCLUDED. BASE CHARGE STORAGE IS MODELED BY FORWARD AND REVERSE TRANSIT TIMES, TF AND TR, AND NONLINEAR DEPLETION LAYER CAPACITANCES WHICH VARY AS THE -1/2 POWER OF JUNCTION VOLTAGE AND ARE DEFINED BY THE PARAMETERS CJE, PE, CJC, AND PC. A CONSTANT COLLECTOR-SUBSTRATE CAPACITANCE, CCS, IS ALSO INCLUDED. THE ENERGY GAP, EG, AFFECTS ONLY THE TEMPERATURE DEPENDENCE OF THE SATURATION CURRENT (SEE PAGE 3).

	NAME	PARAMETER	DEFAULT	TYPICAL
1	BF	FORWARD BETA	100	100
2	BR	REVERSE BETA	1	0.1
3	RB	BASE OHMIC RESISTANCE	0	100
4	RC	COLLECTOR OHMIC RESISTANCE	0	10
5	RE	EMITTER OHMIC RESISTANCE	0	1
6	CCS	COLLECTOR-SUBSTRATE CAPACITANCE	0	2PF
7	TF	FORWARD TRANSIT TIME	0	C.1NS
8	TR	REVERSE TRANSIT TIME	0	10NS
9	CJE	ZERO BIAS B-E JUNCTION CAPACITANCE	0	2PF
10	CJC	ZERO BIAS B-C JUNCTION CAPACITANCE	0	1PF
11	IS	SATURATION CURRENT	1.0E-14	1.0E-14
12	PE	B-E JUNCTION POTENTIAL	1	0.7
13	PC	B-C JUNCTION POTENTIAL	1	0.5
14	VA	EARLY VOLTAGE	INFINITE	50
15	EG	ENERGY GAP	1.11	1.11 FOR SI 0.67 FOR GE

----- GUMMEL-POON BJT MODELS (BOTH NPN AND PNP)

THE INTEGRAL CHARGE MODEL OF GUMMEL AND POON IS A MORE COMPLICATED AND MORE COMPLETE BJT MODEL FOR PROBLEMS WHICH REQUIRE ACCURATE BJT MODELS. THE DC MODEL IS DEFINED BY THE PARAMETERS BFM, C2, IK, AND NE, WHICH DETERMINE THE FORWARD CURRENT GAIN CHARACTERISTICS, BRM, C4, IKR, AND NC, WHICH DETERMINE THE REVERSE CURRENT GAIN CHARACTERISTICS, VA AND VB, WHICH DETERMINE THE OUTPUT CONDUCTANCE FOR FORWARD AND REVERSE REGIONS, AND THE SATURATION CURRENT, IS. THREE OHMIC RESISTANCES, RB, RC, AND RE, ARE INCLUDED. BASE CHARGE STORAGE IS MODELED BY FORWARD AND REVERSE TRANSIT TIMES, TF AND TR, AND NONLINEAR DEPLETION LAYER CAPACITANCES WHICH ARE DETERMINED BY CJE, PE, AND ME FOR THE B-E JUNCTION, AND CJC, PC, AND MC FOR THE B-C JUNCTION. A CONSTANT COLLECTOR-SUBSTRATE CAPACITANCE, CCS, IS ALSO INCLUDED. THE ENERGY GAP, EG, IS INCLUDED AS IN THE SIMPLER BJT MODEL.

	NAME	PARAMETER	DEFAULT	TYPICAL
1	BFM	SAT CURRENT/IDEAL B-E SAT CURRENT	100	100
2	BRM	SAT CURRENT/IDEAL B-C SAT CURRENT	1	0.1
3	RB	BASE OHMIC RESISTANCE	0	100
4	RC	COLLECTOR OHMIC RESISTANCE	0	10
5	RE	EMITTER OHMIC RESISTANCE	0	1
6	CCS	COLLECTOR-SUBSTRATE CAPACITANCE	0	2PF
7	TF	FORWARD TRANSIT TIME	0	0.1NS
8	TR	REVERSE TRANSIT TIME	0	10NS
9	CJE	ZERO BIAS B-E JUNCTION CAPACITANCE	0	2PF
10	CJC	ZERO BIAS B-C JUNCTION CAPACITANCE	0	1PF
11	IS	SATURATION CURRENT	1.0E-14	1.0E-14
12	VA	FORWARD EARLY VOLTAGE	INFINITE	50
13	VB	REVERSE EARLY VOLTAGE	INFINITE	50
14	C2	NONIDEAL B-E SAT CURRENT/SAT CURRENT	0	10.0
15	IK	FORWARD KNEE CURRENT	INFINITE	10MA
16	NE	B-E EMISSION COEFFICIENT	2.0	1.5
17	C4	NONIDEAL B-C SAT CURRENT/SAT CURRENT	0	1.0
18	IKR	REVERSE KNEE CURRENT	INFINITE	100MA
19	NC	B-C EMISSION COEFFICIENT	2.0	1.5
20	PE	B-E JUNCTION POTENTIAL	1.0	0.7
21	ME	B-E GRADING COEFFICIENT	0.5	0.33
22	PC	B-C JUNCTION POTENTIAL	1.0	0.5
23	MC	B-C GRADING COEFFICIENT	0.5	0.33
24	EG	ENERGY GAP	1.11	1.11 FOR SI 0.67 FOR GE

——— JFET MODELS (BOTH N AND P CHANNEL)

THE JFET MODEL IS DERIVED FROM THE FET MODEL OF SHICHMAN AND HODGES. THE DC CHARACTERISTICS ARE DEFINED BY THE PARAMETERS VTO AND BETA, WHICH DETERMINE THE VARIATION OF DRAIN CURRENT WITH GATE VOLTAGE, LAMBDA, WHICH DETERMINES THE OUTPUT CONDUCTANCE, AND IS, THE SATURATION CURRENT OF THE TWO GATE JUNCTIONS. TWO OHMIC RESISTANCES, RD AND RS, ARE INCLUDED. CHARGE STORAGE IS MODELED BY NONLINEAR DEPLETION LAYER CAPACITANCES FOR BOTH GATE JUNCTIONS WHICH VARY AS THE -1/2 POWER OF JUNCTION VOLTAGE AND ARE DEFINED BY THE PARAMETERS CGS, CGD, AND PB.

	NAME	PARAMETER	DEFAULT	TYPICAL
1	VTO	THRESHOLD VOLTAGE	-2.0	-2.0
2	BETA	TRANSCONDUCTANCE PARAMETER	1.0E-4	1.0E-4
3	LAMBDA	CHANNEL LENGTH MODULATION PARAMETER	0	0.01
4	RD	DRAIN OHMIC RESISTANCE	0	100
5	RS	SOURCE OHMIC RESISTANCE	0	100
6	CGS	ZERO BIAS G-S JUNCTION CAPACITANCE	0	2PF
7	CGD	ZERO BIAS G-D JUNCTION CAPACITANCE	0	2PF
8	PB	GATE JUNCTION POTENTIAL	1	0.6
9	IS	GATE JUNCTION SATURATION CURRENT	1.0E-14	1.0E-14

----- MOSFET MODELS (BOTH N AND P CHANNELS)

THE MOSFET MODEL IS ALSO DERIVED FROM THE FET MODEL OF SHICHMAN AND HODGES. THE DC CHARACTERISTICS OF THE MOSFET ARE DEFINED BY THE PARAMETERS VTO, BETA, AND LAMBDA, WHICH ARE IDENTICAL TO THE PARAMETERS FOR THE JFET, PHI AND GAMMA, WHICH DETERMINE THE VARIATION OF THRESHOLD VOLTAGE WITH SUBSTRATE VOLTAGE, AND IS, THE SATURATION CURRENT OF THE TWO SUBSTRATE JUNCTIONS. CHARGE STORAGE IS MODELED BY THREE CONSTANT CAPACITORS, CGS, CGD, AND CGB, AND NONLINEAR DEPLETION LAYER CAPACITANCES FOR BOTH SUBSTRATE JUNCTIONS WHICH VARY AS THE -1/2 POWER OF JUNCTION VOLTAGE AND ARE DETERMINED BY THE PARAMETERS CBD, CBS, AND PB.

	NAME	PARAMETER	DEFAULT	TYPICAL
1	VTO	THRESHOLD VOLTAGE	2.0	2.0
2	PHI	SURFACE POTENTIAL	0.5	0.5
3	BETA	TRANSCONDUCTANCE PARAMETER	1.0E-4	1.0E-4
4	GAMMA	BULK THRESHOLD PARAMETER	0	0.5
5	LAMBDA	CHANNEL LENGTH MODULATION PARAMETER	0	0.01
6	RD	DRAIN OHMIC RESISTANCE	0	100
7	RS	SOURCE OHMIC RESISTANCE	0	100
8	CGS	GATE-SOURCE CAPACITANCE	0	1PF
9	CGD	GATE-DRAIN CAPACITANCE	0	1PF
10	CGB	GATE-BULK CAPACITANCE	0	1PF
11	CBD	ZERO BIAS B-D JUNCTION CAPACITANCE	0	1PF
12	CBS	ZERO BIAS B-S JUNCTION CAPACITANCE	0	1PF
13	PB	BULK JUNCTION POTENTIAL	1	0.6
14	IS	BULK JUNCTION SATURATION CURRENT	1.0E-14	1.0E-14

CONTROL CARDS

***** TITLE CARD

EXAMPLE OP AMP CIRCUIT JOE J STUDENT EECS 241

THIS CARD MUST BE THE FIRST CARD IN THE INPUT DECK. ITS CONTENTS ARE PRINTED VERBATIM AS THE HEADING FOR EACH SECTION OF OUTPUT.

***** .END CARD

GENERAL FORM .END

THIS CARD MUST ALWAYS BE THE LAST CARD IN THE INPUT DECK. NOTE THAT THE PERIOD IS AN INTEGRAL PART OF THE NAME. IF THE .END CARD IS OMITTED, THE NEXT JOB WILL BE READ IN AS PART OF THE JOB MISSING THE .END CARD, AND NEITHER JOB WILL BE RUN SUCCESSFULLY.

***** COMMENT CARD

GENERAL FORM * ANY COMMENTS

EXAMPLE * RF=1K GAIN SHOULD BE 100

THIS CARD IS PRINTED OUT IN THE INPUT LISTING BUT IS OTHERWISE IGNORED.

***** NO PRINT CARD

GENERAL FORM .NP

THIS CARD SUPPRESSES THE SUMMARY OF INPUT DATA THAT IS NORMALLY PRINTED AFTER READING THE INPUT DECK. IT DOES NOT SUPPRESS THE LISTING OF THE INPUT DECK OR ANY ERROR MESSAGES THAT MAY OCCUR.

***** .TEMP CARD

GENERAL FORM .TEMP TE1 TE2 ...

EXAMPLE .TEMP -55.0 25.0 125.0

THIS CARD SPECIFIES THE TEMPERATURES AT WHICH THE CIRCUIT IS TO BE SIMULATED. TE1, TE2, ... ARE THE DIFFERENT TEMPERATURES, IN DEGREES C. A MAXIMUM OF FIVE TEMPERATURES ARE ALLOWED. TEMPERATURES LESS THAN -223.0 DEG C ARE IGNORED.

AD-A055 161 STANFORD UNIV CALIF STANFORD ELECTRONICS LABS F/G 9/2
TECHNIQUES AND APPLICATIONS OF COMPUTER-AIDED CIRCUIT SIMULATIO--ETC(U)
FEB 74 R W DUTTON

UNCLASSIFIED SU-SEL-74-005 NL

2 OF 2
AD A055161

END DATE FILMED 7-78 DDC

END DATE FILMED 7-78 DDC

***** .OUTPUT CARD

```
GENERAL FORM        .OUTPUT VXXXXXX   N+   N-
                    .OUTPUT IXXXXXX VYYYYYY
                    .OUTPUT NOISE

EXAMPLES            .OUTPUT VMIXER 13 27
                    .OUTPUT IBASE1 V17
```

THIS CARD DEFINES AN OUTPUT VARIABLE. FOR VOLTAGE OUTPUTS, THE NAME MUST BEGIN WITH A V, AND N+ AND N- ARE THE POSITIVE AND NEGATIVE NODE OF THE OUTPUT VOLTAGE. FOR CURRENT OUTPUTS, THE OUTPUT NAME MUST BEGIN WITH AN I, AND VYYYYYY IS THE NAME OF THE INDEPENDENT VOLTAGE SOURCE THAT THE CURRENT IS FLOWING IN. POSITIVE CURRENT FLOWS FROM THE POSITIVE NODE, THROUGH THE SOURCE, TO THE NEGATIVE NODE. THE OUTPUT VARIABLE NAME NOISE IS RESERVED FOR THE NOISE ANALYSIS, AND THE OUTPUT NOISE AND EQUIVALENT INPUT NOISE CAN BE PRINTED AND PLOTTED IN THE SAME FASHION AS OTHER OUTPUT VARIABLES.

OUTPUTS CAN BE PRINTED IN TABULAR FORM OR PLOTTED AS LINE PRINTER PLOTS. THERE ARE EIGHT DIFFERENT OPTIONS WHICH CAN BE PRINTED AND/OR PLOTTED:

```
DC      DC TRANSFER CURVE OUTPUT
TR      TRANSIENT ANALYSIS OUTPUT
RE      AC ANALYSIS OUTPUT, REAL PART
IM      AC ANALYSIS OUTPUT, IMAGINARY PART
MA      AC ANALYSIS OUTPUT, MAGNITUDE
PH      AC ANALYSIS OUTPUT, PHASE
OU      NOISE ANALYSIS OUTPUT, TOTAL OUTPUT NOISE VOLTAGE
IN      NOISE ANALYSIS OUTPUT, EQUIVALENT INPUT NOISE
```

AN OUTPUT CAN BE PRINTED OR PLOTTED BY APPENDING THE LETTERS PRINT OR PLOT, FOLLOWED BY ANY COMBINATION OF THE EIGHT OUTPUT OPTIONS, TO THE .OUTPUT CARD:

```
EXAMPLES            .OUTPUT V13 13 0 PRINT MA DC TR
                    .OUTPUT IIN VIN PRINT PH RE DC
                    .OUTPUT VOUT 17 2 PLOT MA TR PRINT DC
                    .OUTPUT I13 V13 PLOT PH DC
                    .OUTPUT VTHREE 3 0 PRINT DC PLOT TRAN
                    .OUTPUT NOISE PRINT IN PLOT OU
```

THE PROGRAM WILL AUTOMATICALLY DETERMINE THE MINIMUM AND MAXIMUM VALUES OF THE OUTPUT VARIABLE AND SCALE THE PLOT TO FIT THESE LIMITS. THE AUTOMATIC SCALING FEATURE CAN BE OVERIDDEN BY SPECIFYING PLOT LIMITS AFTER THE OUTPUT OPTION. THE PLOT LIMITS APPLY ONLY TO THE OPTION THAT THEY FOLLOW.

```
EXAMPLE             .OUTPUT V12 12 0 PLOT MA PH -20 30 TR 0 5
```

IN THIS EXAMPLE, THE PROGRAM WILL DETERMINE LIMITS FOR THE MAGNITUDE PLOT, BUT WILL PLOT THE PHASE BETWEEN -20 DEGREES AND 30 DEGREES, AND WILL PLOT THE TRANSIENT RESPONSE BETWEEN 0 VOLTS AND 5 VOLTS.

***** .DC CARD

GENERAL FORM .DC OP OUTPUT INPUT TC ELNAME VSTART VSTOP VINCR

EXAMPLES
```
.DC OP
.DC TC VIN 0 5 0.5
.DC OP VOUT VIN TC VIN 0 5 0.5
```

FOR THE SMALL SIGNAL TRANSFER FUNCTION, OUTPUT IS THE OUTPUT VARIABLE AND INPUT IS THE INPUT SOURCE. THE PROGRAM WILL COMPUTE THE DC SMALL SIGNAL VALUE OF THE TRANSFER FUNCTION (OUTPUT/INPUT), INPUT IMPEDANCE, AND OUTPUT IMPEDANCE. IF THE TRANSFER FUNCTION VALUE IS NOT DESIRED, OMIT THE OUTPUT AND INPUT SPECIFICATIONS. IF THE DC OPERATING POINT IS NOT DESIRED, OMIT THE LETTERS OP. HOWEVER, A DC OPERATING POINT WILL ALWAYS BE COMPUTED PRIOR TO AN AC SMALL SIGNAL ANALYSIS OR A TRANSIENT ANALYSIS.

FOR TRANSFER CURVES, ELNAME IS THE NAME OF THE VARIABLE SOURCE, VSTART IS THE STARTING SOURCE VALUE, VSTOP IS THE FINAL SOURCE VALUE, AND VINCR IS THE INCREMENT. THE TOTAL NUMBER OF POINTS TO BE COMPUTED CANNOT EXCEED 101. IF A TRANSFER CURVE IS NOT DESIRED, OMIT THE LETTERS TC AND THE TRANSFER CURVE PARAMETERS.

***** .AC CARD

GENERAL FORM
```
.AC DEC ND FSTART FSTOP NOISE OUTPUT INPUT NUMS
.AC OCT NO FSTART FSTOP NOISE OUTPUT INPUT NUMS
.AC LIN NP FSTART FSTOP NOISE OUTPUT INPUT NUMS
```

EXAMPLES
```
.AC DEC 10 1 1CKHZ
.AC DEC 20 1 100KHZ NOISE VOUT VIN 10
.AC DEC 10 1 100MEG NOISE VOUT V21
```

DEC STANDS FOR DECADE VARIATION, AND ND IS THE NUMBER OF POINTS PER DECADE. OCT STANDS FOR OCTAVE VARIATION, AND NO IS THE NUMBER OF POINTS PER OCTAVE. LIN STANDS FOR LINEAR, AND NP IS THE NUMBER OF POINTS. FSTART IS THE STARTING FREQUENCY, AND FSTOP IS THE FINAL FREQUENCY. THE TOTAL NUMBER OF FREQUENCY POINTS TO BE COMPUTED CANNOT EXCEED 101.

FOR NOISE ANALYSIS, OUTPUT IS THE NAME OF A VOLTAGE OUTPUT VARIABLE. THIS OUTPUT, WHICH MUST BE A VOLTAGE, WILL BE USED AS THE SUMMING POINT. INPUT IS THE NAME OF AN INDEPENDENT VOLTAGE OR CURRENT SOURCE. THE TOTAL OUTPUT NOISE IS DIVIDED BY THE TRANSFER FUNCTION (OUTPUT/INPUT) TO OBTAIN THE EQUIVALENT INPUT NOISE LEVEL. NUMS IS THE SUMMARY INTERVAL. AT EVERY NUMS FREQUENCY POINTS, THE INDIVIDUAL CONTRIBUTIONS OF EACH ELEMENT ARE PRINTED OUT. IF NUMS IS OMITTED OR SET TO ZERO, NO SUMMARY PRINTOUT WILL OCCUR. FOR REASONS OF REDUCING PRINTOUT, NUMS SHOULD BE AS LARGE AS POSSIBLE. IF THE NOISE ANALYSIS IS NOT DESIRED, OMIT THE LETTERS NOISE AND THE NOISE ANALYSIS SPECIFICATIONS.

***** .TRAN CARD

GENERAL FORM .TRAN TSTEP TSTOP TSTART FOUR OUTPUT FREQ

EXAMPLES .TRAN 1NS 100NS
.TRAN 1NS 100NS 50ONS
.TRAN 1NS 100NS FOUR VOUT 100MEG

TSTEP IS THE PRINTING INCREMENT BETWEEN TIMEPOINTS, TSTOP IS THE FINAL TIMEPOINT, AND TSTART IS THE INITIAL TIMEPOINT. IF TSTART IS OMITTED, IT IS ASSUMED TO BE ZERO. THE TRANSIENT ANALYSIS ALWAYS BEGINS AT TIME ZERO. IN THE INTERVAL (ZERO, TSTART), THE CIRCUIT IS ANALYZED (TO REACH A STEADY STATE), BUT NO OUTPUTS ARE STORED. IN THE INTERVAL (TSTART, TSTOP), THE CIRCUIT IS ANALYZED AND OUTPUTS ARE STORED. THE NUMBER OF TIMEPOINTS IN THE INTERVAL (ZERO, TSTOP) CANNOT EXCEED 1001, AND THE NUMBER OF TIMEPOINTS IN THE INTERVAL (TSTART, TSTOP) CANNOT EXCEED 101.

FOR FOURIER ANALYSIS, OUTPUT IS THE OUTPUT VARIABLE AND FREQ IS THE FUNDAMENTAL FREQUENCY. THE FOURIER ANALYSIS IS PERFORMED OVER THE INTERVAL (TSTOP-PERIOD,TSTOP), WHERE TSTOP IS THE FINAL TIME SPECIFIED, AND PERIOD IS ONE PERIOD OF THE FUNDAMENTAL FREQUENCY. THE DC COMPONENT AND THE FIRST NINE COMPONENTS ARE DETERMINED. FOR MAXIMUM ACCURACY, THE NUMBER OF PERIODS IN THE INTERVAL (TSTART,TSTOP) SHOULD BE AS SMALL AS POSSIBLE (BUT NEVER LESS THAN ONE). THIS INSURES THAT THE NUMBER OF TIMEPOINTS IN ONE FUNDAMENTAL IS AS LARGE AS POSSIBLE. IF THE FOURIER ANALYSIS IS NOT DESIRED, OMIT THE LETTERS FOUR AND THE FOURIER SPECIFICATIONS.

FOR SOME PROBLEMS, TO AVOID NUMERICAL INSTABILITY IN THE INTEGRATION ALGORITHM, IT MAY BE NECESSARY TO SPECIFY AN INTERNAL TIME STEP WHICH IS SMALLER THAN THE PRINTING INCREMENT (TSTEP). EXAMPLES OF THIS TYPE OF PROBLEM ARE ASTABLE MULTIVIBRATORS, SWEEP CIRCUITS, AND OTHER HIGHLY NONLINEAR CIRCUITS WHICH HAVE WIDELY SEPARATED TIME CONSTANTS. SPICE ALLOWS THE USER TO SEGMENT THE TIME INTERVAL INTO FROM ONE TO FIVE SUBINTERVALS AND SPECIFY A DIFFERENT TIME STEP FOR EACH SUBINTERVAL. THE INTERNAL TIME STEPS AND SUBINTERVAL ENDPOINTS ARE SPECIFIED AFTER THE STARTING TIME (TSTART) AND BEFORE THE FOURIER ANALYSIS OPTIONS:

GENERAL FORM .TRAN TSTEP TSTOP TSTART D1 E1 D2 E2 ... D5 E5 FOUR OUTPUT FREQ

EXAMPLE .TRAN 1NS 100NS 0 0.1NS 10NS 0.5NS 100NS

D1 IS THE FIRST INTERNAL TIMESTEP AND E1 IS THE ENDPOINT OF THE FIRST SUBINTERVAL. D2 IS THE SECOND INTERNAL TIMESTEP AND E2 IS THE ENDPOINT OF THE SECOND SUBINTERVAL, AND SO ON. IN THIS EXAMPLE, THE PROGRAM WILL USE AN INTERNAL TIME STEP OF 0.1NS FOR THE INTERVAL (0,10NS) AND AN INTERNAL TIME STEP OF 0.5NS FOR THE INTERVAL (10NS,100NS). OUTPUT IS STILL STORED EVERY 1NS. THE TOTAL NUMBER OF TIMEPOINTS TO BE COMPUTED CANNOT EXCEED 1001.

EXAMPLE .TRAN 1US 100US 0 0.1US 100US

IN THIS EXAMPLE, THE PROGRAM WILL USE AN INTERNAL TIME STEP OF 0.1US OVER THE ENTIRE TRANSIENT INTERVAL BUT WILL STORE OUTPUT ONLY AT 1US INTERVALS. HENCE, THE PROGRAM STORES AND OUTPUTS EVERY TENTH TIMEPOINT.

EXAMPLE DATA DECKS

THE FOLLOWING DECK DETERMINES THE DC OPERATING POINT AND SMALL SIGNAL TRANSFER FUNCTION OF A SIMPLE DIFFERENTIAL PAIR.

```
SIMPLE DIFFERENTIAL PAIR
VCC 7 0 DC 12
VEE 8 0 DC -12
VIN 1 0
RS1 1 2 1K
RS2 6 0 1K
Q1 3 2 4 MOD1
Q2 5 6 4 MOD1
RC1 7 3 10K
RC2 7 5 10K
RE 4 8 10K
.MODEL MOD1 NPN BF=50 VA=50 IS=1.0E-12 RB=100
.OUT VOUT 5 0
.DC CP VOLT VIN
.END
```

THE FOLLOWING DECK DETERMINES THE DC TRANSFER CURVE AND THE TRANSIENT PULSE RESPONSE OF A SIMPLE RTL INVERTER. THE INPUT IS A PULSE FROM 0 TO 5 VOLTS WITH DELAY, RISE, AND FALL TIMES OF 2NS AND A PULSE WIDTH OF 30NS. THE TRANSIENT INTERVAL IS 0 TO 100NS IN 1NS STEPS.

```
SIMPLE RTL INVERTER
VCC 4 0 DC 5
VIN 1 0 PULSE 0 5 2NS 2NS 2NS 30NS
RB 1 2 10K
Q1 3 2 0 Q1
RC 4 5 1K
.OUTPUT VC 3 0 PRINT DC PLOT TR 0 5
.MODEL Q1 NPN BF=20 RB=100 TF=0.1NS CJC=2PF
.DC TC VIN 0 5 0.1
.TRAN 1NS 100NS
.END
```

THE FOLLOWING DECK DETERMINES THE AC SMALL SIGNAL RESPONSE OF A ONE TRANSISTOR AMPLIFIER OVER THE FREQUENCY RANGE OF 1HZ TO 100MEGHZ.

```
ONE TRANSISTOR AMPLIFIER
VCC 5 0 DC 12
VEE 6 0 DC -12
VIN 1 0 AC 1
RS 1 2 1K
Q1 3 2 4 X33
RC 5 3 500
RE 4 6 1K
CBYPASS 4 0 1UFD
.OUT V3 3 0 PLOT MA PH
.AC DEC 10 1HZ 100MEGHZ
.MODEL X33 NPN BF=30 RB=50 VA=20
.END
```

```
/*
```

External Models in SPICE

SPICE allows external, user-defined models. This feature is particularly convenient when a large circuit includes several identical subcircuits such as operational amplifiers or logic gates. The external model card is defined by a .MODEL card, a set of element cards (which may include any legal SPICE element except reference to another external model), and a .FINIS card. The .MODEL card format is

.MODEL model name X node 1 node 2 . . . node N

The .MODEL card introduces subsequent cards as a definition of an external model. All subsequent cards up to the next .FINIS card are treated as a single definition. These cards are conventional element and device cards and refer to the node numbers that appear in the introducing .MODEL card. Note that with the exception of node zero, the node numbers that are used in the definition of an external model are dummy node numbers and may be given the same number as used within the circuit to be analysed.

Reference to an external model is similar to reference to a built-in model. However, the names of devices described by an external model must begin with the letter X (just as transistors device names must start with the letter Q). Thus one writes

X device name n1 n2 . . . nn model name

The nodes n1 thru nn are actual circuit nodes. They are paired with the node numbers of the external .MODEL definition on the basis of position alone. The first node in the model card is paired with the first node in the device card, the second with the second and so on.

```
VCC 5 0 DC 3.6V
VIN 1 0 PULSE 0 3.6V 0NS 0NS 0NS 40NS
.TRAN 5NS 250NS
.OUT V4 4 0 PLOT TRAN
*
XINV1 1 2 5 INVERTR
XINV2 2 3 5 INVERTR
XINV3 3 4 5 INVERTR
*
.MODEL INVERTR 4 10 30 40
R1 30 20 0 40C
R9 10 20 45C
RC 40 3C 44C
.MODEL A0C NPN BF=30 RB=1 RE=100 RC=100
*   TR=1NS TF=10NS CJC=1PF CJE=2PF
.FINIS
*
.END
```